Arrangement of workstations in pull production lines: A search algorithm approach.

Frankie Tsz-Ki. Fan
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1210
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.
ARRANGEMENT OF WORKSTATIONS IN PULL PRODUCTION LINES: A SEARCH ALGORITHM APPROACH

by

Frankie T. K. Fan

A Thesis Submitted to the Faculty of Graduate Studies and Research Through the Department of Industrial Engineering in Partial Fulfilment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada

1994
The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

### Subject Categories

#### THE HUMANITIES AND SOCIAL SCIENCES

<table>
<thead>
<tr>
<th>Communications and the Arts</th>
<th>Psychology</th>
<th>0525</th>
</tr>
</thead>
<tbody>
<tr>
<td>Art History</td>
<td>Religion</td>
<td>0523</td>
</tr>
<tr>
<td>Cinema</td>
<td>Science</td>
<td>0714</td>
</tr>
<tr>
<td>Drama</td>
<td>Social Sciences</td>
<td>0534</td>
</tr>
<tr>
<td>Fine Arts</td>
<td>Sociology</td>
<td>0335</td>
</tr>
<tr>
<td>Information Science</td>
<td>Special</td>
<td>0529</td>
</tr>
<tr>
<td>Journalism</td>
<td>Teacher Training</td>
<td>0530</td>
</tr>
<tr>
<td>Lib.ry Science</td>
<td>Technology</td>
<td>0710</td>
</tr>
<tr>
<td>Mass Communications</td>
<td>Text and Measurements</td>
<td>0288</td>
</tr>
<tr>
<td>Music</td>
<td>Vocational</td>
<td>0747</td>
</tr>
<tr>
<td>Speech Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Television</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### EDUCATION

| General | 
| Administration | 0515 |
| Adult and Continuing | 0516 |
| Art | 0273 |
| Bilingual and Multicultural | 0282 |
| Business | 0288 |
| Community College | 0275 |
| Curriculum and Instruction | 0277 |
| Early Childhood | 0518 |
| Elementary | 0524 |
| Finance | 0577 |
| Guidance and Counseling | 0519 |
| Health | 0560 |
| History | 0635 |
| History of Education | 0530 |
| Home Economics | 0528 |
| Industrial Arts | 0521 |
| Language and Literature | 0579 |
| Mathematics | 0520 |
| Music | 0522 |
| Philosophy | 0998 |
| Physical Education | 0223 |

#### LANGUAGE, LITERATURE AND LINGUISTICS

| Language | General | 0679 |
| Classical | 0294 |
| Comparative | 0295 |
| Medical | 0297 |
| Modern | 0298 |
| African | 0316 |
| American | 0310 |
| Asian | 0350 |
| Canadian (English) | 0352 |
| Canadian (French) | 0353 |
| English | 0592 |
| European | 0311 |
| Latin American | 0312 |
| Middle Eastern | 0315 |
| Romance | 0523 |
| Slavic and East European | 0314 |

#### PHILOSOPHY, RELIGION AND THEOLOGY

| General | 
| Philosophy | 0422 |
| Religious Studies | 0318 |
| Clergy | 0316 |
| History of Philosophy | 0320 |
| Theology | 0569 |

#### SOCIAL SCIENCES

| American Studies | 0323 |
| Anthropology | 0324 |
| Archæology | 0325 |
| Economics | 0326 |
| Business Administration | 0210 |
| Sociology | 0272 |
| Geography | 0338 |
| Economics | 0501 |
| Agriculture | 0503 |
| Consumer Behavior | 0504 |
| Finance | 0505 |
| History | 0517 |
| Theory | 0511 |
| Folklore | 0365 |
| Geography | 0360 |
| Geology | 0351 |
| History | 0578 |
| General | 

### THE SCIENCES AND ENGINEERING

#### BIOLOGICAL SCIENCES

| Agriculture | 
| General | 0306 |
| Biostatistics | 0303 |
| Botany | 0300 |
| Cell | 
| Embryology | 0317 |
| Ecology | 0318 |
| Entomology | 0239 |
| Microbiology | 0410 |
| Medical Sciences | 0507 |
| Neurosciences | 0317 |
| Oceanography | 0416 |
| Physiology | 0431 |
| Radiation | 0821 |
| Veterinary Science | 0778 |
| Zoology | 0829 |

#### BIOPHYSICS

| General | 0786 |

#### EARTH SCIENCES

| Biogeology | 0425 |
| Geochemistry | 0906 |

#### PHYSICAL SCIENCES

| Geology | 0370 |
| Geophysics | 0372 |
| Hydrology | 0388 |
| Mineralogy | 0413 |
| Paleontology | 0345 |
| Palaeoecology | 0346 |
| Paleolimnology | 0418 |
| Palaeontology | 0985 |
| Physiological Geology | 0357 |
| Physical Oceanography | 0415 |

#### HEALTH AND ENVIRONMENTAL SCIENCES

| Environmental Sciences | 0766 |
| Health Sciences | 
| General | 0566 |
| Audiology | 0567 |
| Dentistry | 0568 |
| General Hospital | 0569 |
| Human Development | 0570 |
| Immunology | 0571 |
| Medical and Surgery | 0572 |
| Mental Health | 0573 |
| Nutrition | 0574 |
| Obstetrics and Gynecology | 0380 |
| Occupational Health and Safety | 0575 |
| Ophthalmology | 0576 |
| Pathology | 0577 |
| Pharmacology | 0578 |
| Physical Therapy | 0579 |
| Public Health | 0580 |
| Radiology | 0581 |
| Recreation | 0582 |

#### Speech Pathology | 0630 |
| Sociology | 0386 |
| Home Economics | 0386 |

#### PHYSICAL SCIENCES

| Pure Sciences | 
| Chemistry | 0656 |
| Agricultural Chemistry | 0649 |
| Analytical Chemistry | 0656 |
| Inorganic Chemistry | 0687 |
| Organic Chemistry | 0688 |
| Physical Chemistry | 0691 |
| Polymer Chemistry | 0694 |
| Radiation | 0734 |
| Mathematics | 0403 |

#### PSYCHOLOGY

| General | 0621 |
| Behavioral | 0624 |
| Clinical | 0625 |
| Developmental | 0626 |
| Experimental | 0627 |
| Industrial | 0628 |
| Personality | 0629 |
| Physiological | 0630 |
| Psychobiology | 0340 |
| Psychometrics | 0631 |
| Social | 0632 |
| 

#### ENGINEERING

| General | 0637 |
| Aerospace | 0538 |
| Agricultural | 0539 |
| Automotive | 0540 |
| Biomedical | 0541 |
| Chemical | 0542 |
| Civil | 0543 |
| Electronics and Electrical Engineering | 0544 |
| Materials Engineering | 0545 |
| Industrial Engineering | 0546 |
| Marine | 0547 |
| Materials Science | 0797 |
| Mechanical Engineering | 0548 |
| Metallurgy | 0763 |
| Mining | 0551 |
| Nuclear | 0552 |
| Packaging | 0549 |
| Petroleum | 0765 |
| Sanitary and Municipal Engineering | 0554 |
| Geotechnology | 0478 |
| Operations Research | 0798 |
| Materials Technology | 0799 |
| Textile Technology | 0904 |

#### PSYCHOLOGY

| General | 
| Behavioral | 0621 |
| Clinical | 0622 |
| Developmental | 0626 |
| Experimental | 0627 |
| Industrial | 0628 |
| Personality | 0629 |
| Physiological | 0630 |
| Psychobiology | 0340 |
| Psychometrics | 0631 |
| Social | 0632 |
I hereby declare that I am the sole author of this thesis.

I authorize the University of Windsor to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Frankie T. K. Fan

I further authorize the University of Windsor to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Frankie T. K. Fan
The University of Windsor requires the signature of all persons using or photocopying this thesis. Please sign below, and give address and date.
ABSTRACT

The purpose of this research is to formulate a mathematical model for a pull production system, and to obtain the optimal, or near-optimal, arrangement of its workstations. The proposed system is a flow shop type production line with a limited number of products. It includes $N$ number of workstations and $N-1$ interstage buffers. Workstations may be allocated to any stage (position) in the production line, and have a different processing rate at each stage. The size of interstage buffers also varies within different arrangements of workstations. The production line is an open system, where all items enter at the first workstation and leave only at the last workstation. It is an unbalanced production line with finite interstage buffers and unreliable workstations in a pull manufacturing environment.

The mathematical model defines the expected total cost of production per item of the finished product. It refers to the allocation of workstations for the proposed system. Costs considered in the model relate to the position of workstations, and include operational, inventory holding, and production shortage costs. Other costs, such as those for materials, and management are not considered because they remain constant for different allocations of workstations. This research uses two search algorithms, genetic algorithm and simulated annealing, to search for a near-optimal allocation of workstations in the production line. These two search algorithms will be applied, and their performances will be compared and studied.
To my parents
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude towards Dr. M. Wang, my thesis advisor, for his support and guidance. I would also like to express my deep gratitude to Dr. S.P. Dutta, Dr. D. Kao, and Dr. S. Taboun, my committee members, for their advice and well wishing.

I wish to thank Mr. Tom William for his technical help and to Ms. Jacque Mummy, secretary of the department, for her assistance and cooperation in the preparation of the thesis.

I also wish to thank my friends, all the graduate students of the Department of Industrial Engineering who have been a constant help and support.

Finally, to my parents, my brother, Timothy, my sisters, Bernice and Wing, I owe the most sincere debt of gratitude for their support, encouragement, and love.
TABLE OF CONTENTS

ABSTRACT vi
DEDICATION vii
ACKNOWLEDGEMENTS viii
LIST OF TABLES xi
LIST OF FIGURES xii

CHAPTER 1: INTRODUCTION 1

1.1 Flow Production Line 1
1.2 Finite Interstage Buffers 1
1.3 Pull Production Line 2
1.4 Unreliable Production Line 3
1.5 Unbalanced Production Line 3
1.6 Performance Measures of Production Line 3
1.7 A Search Algorithm: Genetic Algorithm 4
1.8 Characteristic of Genetic Algorithm 5
1.9 Simulated Annealing 6
1.10 Organization of the Proposed Research 7

CHAPTER 2: LITERATURE REVIEW 9

2.1 Literature Overview 9
2.2 Optimal Buffer Storage Capacity 10
2.3 Unbalanced Production Line 11
2.4 "Bowl" Phenomenon 12
2.5 Unreliable Production Line 15
2.6 Pull Production System . 16
2.7 Genetic Algorithm 19
2.8 Motivation of the Proposed Research 19
2.9 Objectives of the Proposed Research 20

CHAPTER 3: MODEL OF THE PRODUCTION LINE 22

3.1 The Problem 22
3.2 Assumptions and Limitations 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Notations</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>Mathematical Model</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td><strong>CHAPTER 4: APPLICATION OF THE MATHEMATICAL MODEL</strong></td>
<td>52</td>
</tr>
<tr>
<td>4.1</td>
<td>Numerical Examples</td>
<td>52</td>
</tr>
<tr>
<td>4.2</td>
<td>Application of the Mathematical Model</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td><strong>CHAPTER 5: GENETIC ALGORITHM</strong></td>
<td>63</td>
</tr>
<tr>
<td>5.1</td>
<td>Mechanics of Genetic Algorithm</td>
<td>63</td>
</tr>
<tr>
<td>5.2</td>
<td>Model Application</td>
<td>64</td>
</tr>
<tr>
<td>5.3</td>
<td>Application of Genetic Algorithm to the Numerical Example</td>
<td>69</td>
</tr>
<tr>
<td>5.4</td>
<td>Discussion of Result for the Numerical Example of Allocating 5 Workstations</td>
<td>72</td>
</tr>
<tr>
<td>5.5</td>
<td>Other Numerical Examples</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td><strong>CHAPTER 6: SIMULATED ANNEALING</strong></td>
<td>87</td>
</tr>
<tr>
<td>6.1</td>
<td>Mechanics of Simulated Annealing</td>
<td>87</td>
</tr>
<tr>
<td>6.2</td>
<td>Model Application</td>
<td>90</td>
</tr>
<tr>
<td>6.3</td>
<td>Application of Simulated Annealing to the Numerical Example</td>
<td>91</td>
</tr>
<tr>
<td>6.4</td>
<td>Discussion of Result for the Numerical Example of Allocating 5 Workstations</td>
<td>91</td>
</tr>
<tr>
<td>6.5</td>
<td>Other Numerical Examples</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td><strong>CHAPTER 7: CONCLUSION</strong></td>
<td>117</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparison of the Two Search Algorithms</td>
<td>117</td>
</tr>
<tr>
<td>7.2</td>
<td>Conclusion</td>
<td>121</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Work</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td><strong>REFERENCES</strong></td>
<td>124</td>
</tr>
<tr>
<td></td>
<td><strong>APPENDICES</strong></td>
<td>128</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

Table 3.1. Possible State Change. 41
Table 4.1. Processing Rates of Workstations for Example 1. 52
Table 4.2. Failure Rates and Recovery Rates for Workstation. 53
Table 4.3. Inventory Holding Cost and WIP Shortage Cost for Work Stages. 55
Table 4.4. Number of Kanbans and Expected Cost for Stage 2. 58
Table 4.5. Number of Kanbans and Expected Cost for Stage j. 59
Table 4.6. Starving Time $ST_j$ for Workstation at Stage j. 60
Table 4.7. Starving Time $ST_j'$ for Workstation at Stage j. 60
Table 4.8. Allocation of Workstations for Case 1 and Case 2. 62
Table 5.1. Allocation String (A) with 9 Workstations. 65
Table 5.2. Initial Set of Strings for the Numerical Example. 70
Table 5.3. Second Set of Strings for the Numerical Example. 71
Table 5.4. Solution Allocation of Workstation for the Numerical Example. 71
Table 5.5. Number of Kanbans for this Allocation of Workstations. 72
Table 5.6. The Final Set of Strings for the Test with 11 Strings in a Set. 73
Table 5.7. Average Expected Cost and Feasible Solution for Each Trial. 74
Table 5.8. Result of Tests with Different Number of Strings in a Set. 77
Table 5.9. Result of Tests with Different Values of Termination Constant. 80
Table 5.10. Tests Result of Execution Time. 83
Table 5.11. Result of Tests with Different Number of Strings in a Set. 84
Table 5.12. Result of Tests with Different Values of Termination Constant. 84
Table 5.13. Tests Result of Execution Time. 85
Table 5.14. Allocation of Workstations and Number of Kanbans for the Example. 85

Table 6.1. Result of Tests with Different Number of Strings in a Set. 92
Table 6.2. Result of Tests with Different Value of Initial Control Parameter $c_0$. 97

Table 6.3. Result of Tests with Different Values of Termination Constant. 101
Table 6.4. Result of Tests with Different Value of decrement constant $dc$. 104
Table 6.5. Trials Limit $L_t$ for Different Value of decrement constant $dc$. 104
Table 6.6. Result of Tests with Different Value of Initial Trials Limit. 108
Table 6.7. Trials Limit $L_t$ for Different Value of Initial Trials Limit. 109
Table 6.8. Tests Result of Execution Time. 114
Table 6.9. Tests Result of Execution Time. 115
Table 6.10. Allocation of Workstations and Number of Kanbans for the Example. 115

Table 7.1. Average Execution Time for the Four Examples. 118
LIST OF FIGURES

Figure 3.1. Series Queue with Interstage Buffers. 23
Figure 3.2. Flow of Move and Production Kanbans. 25
Figure 5.1. Trial Number vs. Average Cost of Production. 76
Figure 5.2. Number of Strings in a Set vs. Percentage of Duplication of String. 78
Figure 5.3. Number of Strings in a Set vs. Execution Time. 79
Figure 5.4. Termination Constant vs. Optimal Achieve Probability. 81
Figure 5.5. Termination Constant vs. Execution Time. 82
Figure 6.1. Number of Strings in a Set vs. Optimal Achieve Probability. 94
Figure 6.2. Number of Strings in a Set vs. Total Number of Trials. 95
Figure 6.3. Number of Strings in a Set vs. Execution Time. 96
Figure 6.4. Value of Initial Control Parameter vs. Probability of Strings Accepted. 98
Figure 6.5. Value of Initial Control Parameter vs. Total Number of Trials. 99
Figure 6.6. Value of Initial Control Parameter vs. Execution Time. 100
Figure 6.7. Termination Constant vs. Optimal Achieve Probability. 102
Figure 6.8. Termination Constant vs. Execution Time. 103
Figure 6.9. Value of Decrement Constant vs. Probability of Strings Accepted. 105
Figure 6.10. Value of Decrement Constant vs. Execution Time. 106
Figure 6.11. Trial Number vs. Trial Limit for Different Value of Decrement Constant. 107
Figure 6.12. Initial Trial Limit vs. Optimal Achieve Probability. 110
Figure 6.13. Initial Trial Limit vs. Total Number of Trials. 111
Figure 6.14. Initial Trial Limit vs. Execution Time. 112
Figure 6.15. Trial Number vs. Trial Limit for Different Value of Initial Trial Limit. 113
Figure 7.1. Number of Workstations vs. Execution Time. 119
CHAPTER 1

INTRODUCTION

1.1 Flow Production Line

A serial flow production line is a fundamental type of manufacturing system. It consists of a set of machines arranged in a contiguous series. Initially, materials or parts are processed through the first workstation in the system. They are then forwarded sequentially from one workstation to the next. Each workstation is responsible for processing unfinished items received from the preceding workstation. Finally, the finished items leave the production line after processing at the last workstation.

1.2 Finite Interstage Buffers

There are interstage buffers between workstations for the storage of work-in-process (WIP) inventory. Buffer capacity is assumed to be finite in this research, and will be discussed in the next section. If a buffer is full and the preceding workstation completes an item, that item is forced to stay in the workstation and blocking occurs. On the other hand, if a workstation is ready to accept an item and its input buffer is empty, it must stay idle until an item enters the buffer. This phenomenon is known as "starving". The first workstation is never starved, assuming there is an infinite supply of resources or preprocessed items. Similarly, the last workstation is never blocked, assuming there is an infinite amount of output space.
A significant amount of research has been published on the subject of determining the optimal buffer capacity for the flow production line since the late 1950's, and is stated in the literature review of the next chapter. Research continues to develop with the development of the modern manufacturing system, such as the unbalanced production line approach and the pull-type manufacturing system (just-in-time system).

1.3 Pull Production Line

Recently, Japan's just-in-time (JIT) pull manufacturing system has been very successful. The benefit of potential inventory reductions and productivity gains with the application of the JIT system has attracted the attention of North American companies. This system is a new environment for the flow production system. Its major objective is to reduce the work-in-process inventory to an absolute minimal level. Since the capacity of interstage buffers is reduced to achieve this level, the reliability of the machines becomes an important consideration in the design of manufacturing system.

1.4 Unreliable Production Line

The workstations, or machines, are unreliable in the model discussed here. They fail at random and remain so for a random period of time while they are under repair. If a machine breaks down, the preceding workstation can operate until its output buffer is full. Similarly, the succeeding workstation can still operate until its corresponding input buffer is empty. In this way, the buffers can offset the effects of breakdowns and increase the productivity of the production line.
1.5 Unbalanced Production Line

In early studies of the production line, the objective was balancing the production line in terms of the mean operation time. However, in the research of Davis (1966), it was proposed that the production rate be increased and workstation idle time be reduced by unbalancing a production line. Further research shows that this approach is better for a production line with high variability in operation times, low interstage buffer capacity, and large number of workstations. This will be discussed in the literature review.

1.6 Performance Measures of Production Line

The most common performance measure variable of the production line is its throughput rate. The mean time to starving (MTTS) is also a good evaluation of performance for an unreliable production line. It is the time that a production line could operate without starving after a workstation breaks down. System reliability is a way to evaluate unreliable production lines. Another common performance measure is production cost, which could consider all the above performance measures.

Several factors may affect performance. As an illustration, the arrangement of workstations could significantly influence the efficiency of the production line. Other factors include the capacity of the interstage buffers, the production rate, and the reliability of the workstations. As well, performance also depends on the availability of resources to repair a malfunctioning workstation.
1.7 A Search Algorithm: Genetic Algorithm

In the proposed research, the objective is to obtain the optimal arrangement of workstations to increase the efficiency of the production line. Genetic algorithm is used as a search algorithm because the computation is simple but powerful in the pursuit for improvement. Its mechanics involve, simply, random number generation, copying strings and swapping partial strings. The process performs simply and powerfully, which are two major attractions of the genetic algorithm.

Genetic algorithm was first introduced by John Holland (1975), his colleagues, and his students, at the University of Michigan. Their research covered the areas of both natural and artificial systems science. The abstract rigorously explains the adaptive processes of the natural system. Moreover, the development of artificial system software retains the important mechanisms of the natural system.

"Genetic algorithm is based on the mechanics of natural selection and natural genetics. This procedure uses random choice as a tool to guide a highly exploitative search through a coding of the parameter space. It combines the survival of fittest over the string structure with a randomized information exchange to form a search algorithm and the innovative flair of human search. This algorithm creates a new set of artificial strings in each generation, by using all the bits and pieces of the old strings' fittest. Genetic algorithm efficiently exploits the historical information to predict new search points with an expected improved performance.

The main theme of the research on genetic algorithms is robustness, the balance of efficiency and effectiveness required for survival in many different environments. The natural system has good robust performance; the secrets of adaptation and survival are clearly revealed in the study of biological examples. With those studies, genetic algorithm is theoretically and empirically proven to provide robust performance in
complex space.\textsuperscript{1}

Since the first monograph in genetic algorithm, "Adaptation in Natural and Artificial Systems", by John Holland, many more papers and dissertations in this area have been presented. Today, the simplicity and power of genetic algorithm is applied widely in business, science, and engineering in the search for improvement. As well, this algorithm has no fundamental restriction in the assumption of search space.

1.8 Characteristic of Genetic Algorithm

A comparison of genetic algorithm with traditional search methods shows that it has a robust performance because it differs from most normal optimization and search procedures in four fundamental ways:

- Genetic algorithm works with a set of coding parameters, not simply with one parameter.
- Genetic algorithm searches from a population of points, not from only a single point.
- Genetic algorithm uses payoff (objective function) information, not derivatives or other auxiliary knowledge.
- Genetic algorithm uses probabilistic transition rules, not deterministic rules.

Genetic algorithm requires the natural parameter set of the optimization problem to code

as a finite-length string over some finite alphabet. The algorithm deals with the input set of coding to predict the payoff function. The process of transforming the input to output does not need to be considered. Therefore, the transformation process could be considered as a black box.

Many ordinary optimization methods search from a single point in the decision space to the next using some transition rules. Such approaches would easily allocate a false peak in a multimodal (many-peaked) search space. However, a genetic algorithm seeks from a rich database of points (a population of strings). Thus, its probability of locating a false peak is greatly reduced, compared with traditional methods.

Most traditional techniques require much auxiliary information to work appropriately. A genetic algorithm approach does not need any auxiliary information. The only information needed is the payoff value (objective function value) associated with each individual string.

Unlike many other methods, which use deterministic rule, the genetic algorithm uses probabilistic transition rules to guide its search direction. It uses random choice as a tool to guide the search, approaching regions of the search space with likely improvement.

1.9 Simulated Annealing

"Annealing is a physical process, in which a solid is heat until it melts, then cooled until it crystallizes into a state with perfect lattice. The free energy of the solid is minimized
in this process. The cooling must be done carefully, to prevent trapping in locally optimal lattice structures with crystal imperfections. 2

In the early 1980's, Kirkpatrick, Gelatt and Vecchi (1982, 1983), and independently, Cerny (1985), introduced the application of annealing to combinatorial optimization. These concepts were based on a strong analogy between annealing and the problem of solving large combinatorial optimization problems.

The combinatorial optimization problem is formulated as one of determining a solution with minimal cost among a potentially very large number of solutions in a solution space. It is done by establishing a correspondence between the cost function and the free energy, and between the solutions and the physical states. A solution method could be introduced to the field of combinatorial optimization based on a simulation of the physical annealing process. Known as simulated annealing, the advantage of this method is its general applicability, and its ability to obtain solutions arbitrarily close to optimum.

1.10 Organization of the Proposed Research

This research is organized as follows: A review of the modelling approaches to the design of the flow production line is discussed in Chapter 2. Motivation and objectives of the proposed research are also included in Chapter 2. Production line models are formulated in Chapter 3. A numerical example of the mathematical model application is shown in Chapter 4. Genetic algorithm, is presented and implemented in the problem

---

of production line arrangement in Chapter 5. Simulated annealing, the second search
algorithm, is presented and implemented in the problem in Chapter 6. The computational
results are discussed in Chapter 7. Finally, the conclusion and suggestions for further
research are provided also in Chapter 7.
CHAPTER 2

LITERATURE REVIEW

2.1 Literature Overview

Production line modelling and design problems have been presented quite extensively by many researchers. In 1956, Hunt used the queuing theory approach to simulate small flow lines and examine their operating characteristics. Many researchers extended Hunt’s idea. Barton (1962) and Davis (1966) applied computer simulation to study more complex flow line systems. The development of the modern manufacturing system has inspired further research. Buzacott and Hanifin (1978) presented a concise review and comparison of some earlier models.

In most of the early research, the objective was to balance the production line and no breakdowns were assumed. However, Davis (1966) reported some gains in the application of an unbalanced production line. This discovery motivated the research by Payne, Slack, and Wild (1972), and Sadowski and Medeiros (1979).

In the extension of the unbalanced production line, a well-known approach is the "bowl" phenomenon. This phenomenon was developed by Davis (1966) and further approved by Carnall and Wild (1976), Rao (1976), El-Rayah (1979a, b), Magazine and Silver (1978), Muth and Alkaff (1987), and Hillier and So (1993), for different types of production lines. Later, Gershwin and Schick (1983), D’Angelo et al. (1988), Hillier
and So (1991), and Glassey and Hong (1993) suggested the importance of considering
the unreliability of workstations in a production system.

Finally, the first study of production line arrangement in the just-in-time environment
was performed by Villeda, Dudek, and Smith (1988). The concept was extended by
Wang and Wang (1990), Park and Steudel (1991), and Albino, Carella, and Okogbaa

2.2 Optimal Buffer Storage Capacity

Anderson and Moodie (1969) used simulations (GPSS-III language) to estimate the
coefficients of an optimal solution to the problem of buffer capacity. They determined
the buffer capacity for two models of steady-state production systems with varying
lengths, to examine the influence of the buffer inventories on the operation costs of a
production line. The research found that minimum cost occurred when the buffer
capacity maintained the best steady level for the entire production period. The study also
discovered that there was no need to control the inventory level during the transient
period.

Numerous others have performed research to determine optimal buffer storage capacity.
A recent paper was presented by Park (1993). He addressed the unique characteristics
involved in buffer design, and discussed drawbacks in the application of traditional
optimization methods. He suggested an efficient two-phase heuristic approach, based on
the dimension reduction strategy and the buffer utilization-based beam search method. Park's research did not, however, consider the effects of unreliable workstations and the JIT environment.

2.3 Unbalanced Production Line

The objective of early studies was to balance the mean production rate of the system. As mentioned, Davis' (1966) report of gains in an unbalanced production line initiated further work in this area.

Payne, Slack, and Wild (1972) studied flow production lines with "balanced" and "unbalanced", independent, normally distributed workstation service times. They compared idle time and maximum queue length of the computer simulation for both systems. The effect of "imbalance" in workstations' mean service time, variability of processing rate, idle time, maximum queue length and output were also investigated. The results showed that a reduction of workstation idle time was obtained by positioning the workstation at the end of the line, which was either with a greater mean of service time or greater variability. The study also stated that perfect line balance was not compatible with maximum flow line utilization. Furthermore, the optimum buffer inventory capacity depended upon the position of the stock holding in the flow production system and the operating condition.

Sadowski and Medeiros (1979) proposed the "One Worker at A Time" (OWAAT)
algorithm, to assess the effect of variability in selected factors on the labour-involved, unbalanced production line. The performance measures were buffer capacity, worker efficiency, absenteeism, and penalty for shortage of supply. Although a high degree of variability was obtained in the results, the buffer capacity, penalty for shortage, and efficiency factors were significant to the average work-in-process inventory cost. Absenteeism did not influence production line performance. The research also revealed that the work-in-process inventory was not a desired concern with respect to inventory cost, unless the penalty of shortage cost was extremely high.

Hopp and Simon (1993) studied an assembly-like flow system where the assembly machine was fed from the storage buffers of two input machines. This was a case of multiple workstations in one stage. They derived an approximate expression for the steady state average throughput of the production system and used it to optimize buffer capacity. Their research mentioned the unreliable workstation. However, it did not cover the JIT environment.

2.4 "Bowl" Phenomenon

The "bowl" phenomenon is the assignment of workstations with comparatively lower operation time to the middle of the production line. Hillier and Boling (1966) first discovered this occurrence during the investigation of an approximate analytical approach in three and four stages production lines. Other researcher confirmed the bowl phenomenon in different cases of production lines.
Carnall and Wild (1976) continued the research of Payne, Slack, and Wild (1972). They considered the effect of unbalancing in a production line with variable operation times. The study indicated that variability in operation time did affect production rate and average idle time of the production system, but not buffer capacity. Although, this research assumed no breakdown, it did agree with the bowl phenomenon established by Hillier and Boling (1966). During the same time, Rao (1976) proposed that unbalancing could lead to improvements in production rate, if the variability of the operation times was different for each workstation. This research also confirmed with the bowl phenomenon.

El-Rayah (1979a) compared three different arrangements of workstations, in three, four and twelve workstations production lines. These arrangements were the bowl phenomenon, a monotonic increasing order, and an alternating low-high arrangement. The results illustrated that the bowl phenomenon has a better output rate. The research also showed that the unbalanced production line had a better output rate than the balanced one, due to:

- greater variability in operation times
- smaller interstage buffer capacity
- larger number of workstations in the production line.

These factors increased the probability of blocking and starving in workstations. They also reduced the output rate for the balanced production line. This research did not consider breakdowns.
El-Rayah (1979b) extended his research to investigate the effect of inequality of interstage buffer capacities and operation time variability on the unbalanced production line (according to the bowl phenomenon). The result showed that inequality in operation times variability affected the output rate as the inequality in operation times. There was very little influence on the average number of work-in-process inventories. On the other hand, the inequality of interstage buffer capacities did affect the number of WIP and only slightly affect the output rate.

Magazine and Silver (1978) developed the heuristics to determine the output rate and workstation allocation in the flow production line. The results corresponded with Hillier and Boling's results (1966). Their research also showed that output rates could be augmented with an increase in buffer capacity, a decrease in workstations, or through unbalancing. The study also noted that the greatest effect of unbalancing occurred with an intermediate number of workstations with small buffer capacities. If the number of workstations was small, changing it would greatly affect the output rate.

Muth and Alkaff (1987) reviewed the application of the bowl phenomenon to analyze serial production lines. They presented the concept in several distribution models of a three workstation production line. The models were used to compute the throughput rate of unbalanced lines, in which the sum of the mean service times was constant. Their results are shown as contour plots of constant throughput rate. They used the models to present the research of Hillier and Boling (1966) and Rao (1976).
Hillier and So (1993) revised their early research (1966, 1979) in bowl phenomenon, by extending the numerical results for it application to an unbalanced production line with variable processing times. They suggested some guidelines and information for extrapolating the results, to define the optimal allocation of work for some larger production lines, which could not be solved through exact solution methods. However, they did not address unreliable workstations and pull manufacturing systems.

2.5 Unreliable Production Line

Gershwin and Schick (1983) modelled a flow production line with three unreliable workstations and a finite buffer capacity in the Markov chain. The system states included the operation states of the workstations and the inventory level in the interstage buffer. Consequently, their steady state probabilities were sought. This algorithm suggested a general approach for solving large scale, structured Markov chain problems.

In another approach, D'Angelo, et al. (1988) constructed an event-driven model for an unreliable production line with finite buffer capacity. The event was the change of system states, including the operation states of the workstations and the inventory level in the interstage buffer. This model utilized simulation and some elementary analysis as well. Excessive computation time and piece-by-piece processing of the model was avoided. Thus, this model was unique, and capable of analyzing a large model in short time.
Hillier and So (1991) studied the effects of machine breakdowns and interstage buffer capacity on performance of the production line. A simple heuristic method was used to determine the optimal buffer capacity, which offset the effect of machine breakdown in the unreliable production system.

Glassey and Hong (1993) studied the behaviour of an unreliable $N$ th stage with $(N-1)$ interstage storage buffers, developing the decomposition approach introduced by Gershwin (1987). The $N$ stage production line is decomposed into $(N-1)$ aggregate two stage lines, for which analytical solutions are available. This concept was developed through the steady state behaviour and the unreliable workstation. They also compared the method with simulation, to find that their approach was efficient in computation and performed quite well. This research did not study the application of the pull production system.

2.6 Pull Production System

In the late 1980's, North America and Europe had an increase in utilization of the just-in-time system. Hence, et al. (1988) began to examine the flow production line in the JIT environment. This research considered different arrangements of workstations. The arrangement with the best output rate was the unbalancing method of descending order of variable operation time. The research showed that the unbalanced production line was always superior to the perfectly balanced one. The improvement in output rate of the balanced production line increased the variability of operation times in the last
workstation, and decreased the system's interstage buffer capacity. However, this research did not account for workstations breakdown.

Park and Steudel (1991) developed a model to determine the job throughput rates for manufacturing flow line workcells with finite buffers. The research considered four factors:

- sequential workstations,
- finite buffer capacity,
- piece-by-piece continuous flow processing,
- separate consideration between setup time and processing time.

They also compared this model with a deterministic simulation model. The conclusion showed that it provided an accurate result in the simulation and required minimal computation time. It was more efficient than the deterministic simulation model. This model also did not consider workstations breakdown.

Albino, Carella, and Okogbaa (1992) presented a discrete-event simulation model. This model evaluated the performance measures of a kanban controlled JIT manufacturing flow line, with unreliable machines and different maintenance policies. The measures were throughput rate, average number of work-in-process inventories, and backorder level. The maintenance policies were only corrective maintenance and corrective plus preventive maintenance. The simulation showed that the corrective plus preventive maintenance policy had better performance. While the arrangement of the workstations
was not observed in this study, the model could be developed to consider it.

Wang and Wang (1990) developed a mathematical model to define the optimal number of kanbans for a JIT flow production line. The number of kanbans between two adjacent workstations decided the inventory level of that pair of workstations. By minimizing the work-in-process inventory level, a better JIT system could be achieved, with process improvement and reduced process variability. Moreover, it could lead to minimal WIP, or even zero inventory level. The model was solved by using a Markov process approach, using the demand rate of finished products as the departure rate, and the production rate of workstation as the arrival rate. Finally, Wangs’ research did consider the case of unreliable workstations. Generally speaking, the presenting research is an extension of Wangs’ research paper. In their research, the optimal number of kanbans was obtained. After that, this information is used to determine the optimal arrangement of workstations.

Berkley (1992) reviewed literature on kanban production systems. He discussed the setting of kanban number, performance measures, material-handling frequencies, and batch-sizing. He also compared the kanban production system with conventional system. Abdou and Dutta (1993) used systematic simulation approach to find the optimum interarrival time of the material handler, the optimum container capacity and the optimum number of kanbans.
2.7 Genetic Algorithm

Goldbery (1987) reviewed the genetic algorithm, stating its history, theory, development and application. Biegel and Davern (1990) applied the genetic algorithm to a flow job shop scheduling. Genetic algorithm solved the problem more efficiently than the deterministic search approach, and computation time was minimal. The research showed that the genetic algorithm was a simple but powerful search algorithm.

2.8 Motivation for the Proposed Research

In summary, the research of Payne, Slack and Wild (1972), and Sadowski and Medeiros (1979), was concerned with the performance of the unbalanced production line. The articles of Carnall and Wild (1976), Rao (1976), El-Rayah (1979a, b), Magazine and Silver (1978), Muth and Alkaff (1987), and Hillier and So (1993) showed agreement of the bowl phenomenon, developed by Hillier and Boling (1966) in the traditional push production system. Gershwin and Schick (1983), D'Angelo, Caramanis, Finger, Mavretic, Phillis, and Ramsden (1988), Hillier and So (1991), and Hopp and Simon (1993) proposed the importance of considering workstation reliability in the flow production line.

Finally, the research paper of Villeda, Dudek, and Smith (1988) defined the arrangement of workstations with the best output rate. This arrangement was the unbalanced production line with descending order of variable operation time, and a finite buffer in the JIT environment. The result did not consider the probability of workstation
breakdown. However, it was totally different from the bowl phenomenon of traditional push production environment. This study induced the pursuit of the best arrangement of workstations for the pull production line with unreliable machines.

Wang and Wang (1990) presented a research to determine the optimal number of kanbans (buffer capacity) for the unreliable JIT production line with finite buffer capacity. With the optimal number of kanbans as a basis, it is able to determine the optimal allocation of workstations.

In the proposed research, the manufacturing system employed is an unbalanced, unreliable pull production line with finite interstage buffers. The optimal, or near-optimal, allocation of workstations in this system will be obtained by two search approaches, genetic algorithm and simulated annealing.

2.9 Objectives of the Proposed Research

The objectives of the proposed research are:

1. To formulate the mathematical model of the unbalanced production line with finite interstage buffers and unreliable workstations in the pull manufacturing environment.

2. To determine the optimal, or near optimal, allocation of workstations for the proposed manufacturing system by the two search algorithm approaches, genetic algorithm and simulated annealing.
3. To study and compare the results obtained by the two search approaches.
CHAPTER 3
MODEL OF THE PRODUCTION LINE

In this chapter, a model is formulated for the unbalanced production line with finite
interstage buffers and unreliable workstations in the pull environment. The model
formulates the production of a single specific product type. This production line is a
flow shop type with a limited number of products in the system.

3.1 The Problem

The objective of this research is to determine the optimal, or near-optimal, arrangement
of workstations for the production system. The production line under consideration is
a pull manufacturing system designed to produce a single product. A production system
which manufactures multiple products could be examined in a future extension of this
research. This production system includes $N$ workstations in sequence, and $N-1$ finite
interstage buffers, as shown in Figure 3.1. It is an open system, where items enter at
the first workstation and leave at the last workstation. All items proceed through all
workstations. The production system is in tandem configuration, with only one
workstation in each work stage (position). This situation is most commonly found in a
flow machine shop, where parts are machined at one station and then transported to
another for the next operation. The case of multiple stations could be studied as a
development of this research.
Figure 3.1. Series Queue with Interstage Buffers.
In the proposed system, each workstation can be allocated to a different position in the line, to operate a different job with a different processing rate. Additionally, all $N$ workstations will be allocated in $N$ positions. All workstations are assumed to be able to perform multiple tasks, and to be allocated in different stages of the process. However, there may be some restrictions. Some workstations may not perform particular jobs, or should be allocated before or after other workstations. A certain workstation may also be restricted to a specific position to perform a particular job.

This model uses the approach of an unbalanced production line. El-Rayah (1979a) stated that this approach is better for a system with low interstage buffer capacity, large number of workstations, and high variability in operation times.

The pull manufacturing system uses two types of kanbans, the production kanban and the move kanban, as shown in Figure 3.2. The production kanban controls the system's production, while the move kanban controls the movement of parts between workstations. For the production activity in stage $i$ ($1 \leq i \leq N$), a container with an item inside, and a required item accompanied by a move kanban, both enter stage $i$ from the preceding stage $i-1$. Then, an available production kanban at stage $i$ replaces the move kanban, which is sent to the move kanban post. Simultaneously, the item accompanied by a production kanban begins to process in stage $i$. When the item is demanded by the succeeding stage $i+1$ (as indicated by a move kanban from stage $i+1$), the production kanban leaves the container: and returns to the production kanban post at stage $i$. The
Fig. 3.2. Flow of Move and Production Kanbans.
move kanban from stage \( i+1 \) replaces the production kanban in the container at stage \( i \). Then the move kanban travels with the container, with the item inside, to the succeeding stage \((i+1)\). Thus, the production kanban moves inside a stage, while the move kanban moves between stages. It is assumed that there is only one move kanban between two workstations. However, the number of production kanbans could be more than one (Berkley, 1992). The optimal number of production kanbans could be determined by the mathematical model shown in a later section. For this research, the interstage buffer capacity is the number of production kanbans. It is minimized in order to reduce the WIP inventory.

The number of fulfilled buffers between workstations is considered to be stochastic. The arrival rate of the Markov process is the production rate of the preceding workstation. The departure rate is the demand rate of finished products, reflecting the characteristics of a pull production system. In such a system, the demand rate for finished products is sent to preceding workstations as the local demand rate. The mathematical model assumes the production and demand rates of each workstation to be exponentially independent. Solberg’s research, (Solberg 1977, 1981, Solberg and Nof 1980), stated that exponential distribution was much more aligned to real-world production. Since exponential distribution appropriately describes the production rate, it was often used in previous studies.

In this mathematical model, the workstations are unreliable which breakdown according
to a failure rate. Failure rate is the number of workstations break down over a period of time. The model assumes the workstations' failure rate to be exponential distribution, and that processing is arrested during repair. Recovery rate is the number of broken workstations the repair personnel could repair over a period of time. The model also considers the recovery rate of a broken workstation to be exponential distribution. These premises are suggested by Gerahwin (1983). If a machine breaks down, preceding workstations can continue to operate until their output buffers are full. Similarly, succeeding workstations can continue until their corresponding input buffers are empty. As mentioned, in this way the buffers offset the effect of breakdown and increase the productivity of the line.

3.2 Assumptions and Limitations

The mathematical model has the following assumptions and limitations:

1. There is an infinite source at the beginning of the production line, and an infinite storage capacity at the end. Thus, the first machine is never starved and the last one is never blocked.

2. The transportation time of an item, from one workstation to the next, is negligible.

3. Loading and unloading times of workstations are included in the processing rate.

4. There is no consideration of scrap and rework items.

5. Each item to enter a workstation must be completely processed before leaving it.

6. Repair personnel stops and repairs a broken workstation only after a process is
complete and the item has left. Therefore, breakdown time begins immediately following a processing stage.

7. The mean processing time of a single item at any workstation is an independent random variable with an Poisson distribution (verified in the previous section). The processing time could be considered for one part, or for \( n \) parts in a batch (container).

8. Failure and recovery rates of workstations are independent random variables follow exponential distribution.

9. Breakdown cannot occur at a starved or blocked workstation.

3.3 Notations

The following subscripts and notations are used in the mathematical model, and defined later in this chapter.

Subscripts

\( i \) – index of workstation number, \( i = 1, \ldots, n \).

\( j \) – index of work stage (position), \( j = 1, \ldots, n \).

\( k \) – index of work stage (position), for cases requiring a second index, \( k = 1, \ldots, n \).

\( l \) – index of kanban state for the buffer, \( l = 1, \ldots, K \).
Notations

$TC$ $-$ Total cost of production per item.

$CI$ $-$ Operation cost per unit time during steady state, which includes labour, machine, maintenance, etc.

$C2_{ij}$ $-$ Inventory holding cost plus shortage cost of workstation at stage $j$ per unit time.

$C3$ $-$ Shortage cost per unit time of finished product demanded for the production system.

$PW_i$ $-$ Position of workstation $i$.

$P_{ij}$ $-$ Assignment variable of workstation $i$ to position $j$.

$N$ $-$ Number of workstations (assume the production line uses all workstations, so the number of workstations is equal to the number of positions).

$D$ $-$ Demand rate of finished products.

$F_i$ $-$ Failure rate of workstation $i$, which is assumed to be in Poisson distribution.

$R_i$ $-$ Recovery rate of workstation $i$, which is assumed to be in Poisson distribution.

$\mu_{ij}$ $-$ Mean processing rate of workstation $i$ in stage $j$ per unit time, which is assumed to be in Poisson distribution.

$HC_{ij}$ $-$ Inventory holding cost per item per unit time, for workstation at stage $j$. 

29
$SC_{sj}$ — Shortage cost of WIP item per unit time, for workstation at stage $j$.

$Fs_{sj}$ — Failure rate of workstation in stage $j$.

$Rs_{sj}$ — Recovery rate of workstation in stage $j$.

$\mu s_{sj}$ — Mean processing rate of workstation in stage $j$.

$Ks_{sj}$ — Number of move kanbans circulating between the workstations in stage $j$ -1 and $j$.

$I_{sj}$ — Inventory level in the buffer before the workstation in stage $j$, during steady state.

$ST_{sj}$ — Starving time of the system caused by the breakdown of workstation in stage $j$.

$THR$ — Throughput rate of the system during steady state.

$MST$ — Mean starving time of the system.

$E$ — State space of the Markov process, which describes the number of fulfilled containers (kanbans) sent to succeeding workstations.

$Q_k$ — Transition matrix of the Markov process between workstations with $K$ kanbans.

$\pi_{k}(n)$ — Limiting distribution of state $n$ of the Markov process with $K$ kanbans.

$C_a$ — Total cost factor of staying in state $n$ per unit time.

$II$ — Status indicator for the preceding workstation. The indicator is 1 when the workstation is up and the indicator is 0 when the workstation is broken.
down.

I2 — Status indicator for the succeeding workstation. It works in the same way as I1 above.

3.4 Mathematical Model

The mathematical model is used to evaluate different arrangement of workstations to determine the optimal or near-optimal allocation. It uses the total cost of production per item \( (TC) \) as the performance measure of the system. The costs considered in this model are those related to the allocation of the workstations. The total cost has three components:

- operation cost such as direct labour cost, machining cost, maintenance cost, etc. per unit time \( (C1) \)
- inventory holding cost plus shortage cost of WIP parts for each workstation per unit time \( (C2) \)
- shortage cost of finished products for the production system per unit time \( (C3) \).

Other costs (materials, factory rent, etc.) are not considered because they do not affect the allocation of the workstations in the system. The procedure for determining the total cost of production per item \( (TC) \) will be presented in the following steps.

Step 1: Generate the assignment variable matrix

This model attempts to determine the total cost of production per item \( (TC) \) for a specific
arrangement of workstations. The given variables are the number of workstations $N$ (assume that the system uses all workstations, so the number of workstations equals the number of positions), and all the $N$ given positions of workstations ($PW_i$) for workstation $i$ allocated in position $PW_i$. For example, if there are five workstations and five positions ($N = 5$), and $PW = (1,2,3,4,5)$, for workstation 1 assigned to position 1, ($PW_1 = 1$), and workstation 2 assigned to position 2, ($PW_2 = 2$), etc. The assignment variable of workstations ($P_{ij}$) can be determined by the following equation.

$$P_{ij} = \begin{cases} 1 & \text{for } PW_i = j \\ 0 & \text{otherwise} \end{cases} \text{ for } i = 1 \text{ to } N, \ j = 1 \text{ to } N$$

(1)

In equation (1), the assignment variable of the workstation ($P_{ij}$) is equal to 1 for the assignment of workstation $i$ to position $j$. Otherwise, the assignment variable of the workstation ($P_{ij}$) is equal to zero.

**Step 2: Check for illegal allocation**

The assignment variable matrix defined in Step 1 has to be checked by the following two assignment constraints for illegal allocation.

- only one workstation is assigned to each position

$$\sum_{j=1}^{N} P_{ij} = 1 \quad \text{for } j = 1 \text{ to } N$$

(2)

- each workstation is assigned to only one position

$$\sum_{i=1}^{N} P_{ij} = 1 \quad \text{for } i = 1 \text{ to } N$$

(3)
There are also some allocation restriction:

- no allocation restriction (for workstation \( x \) cannot be allocated to position \( y \) )

\[
P_{xy} = 0
\]  
(4)

- allocation restriction (for workstation \( x \) must be allocated to position \( y \) )

\[
P_{xy} = 1
\]  
(5)

- Preceding constraint / succeeding constraint (for workstation \( x \) must be allocated in a position before workstation \( y \) )

\[
PW_x < PW_y
\]  
(6)

For some cases the allocation of workstations does not agree with the assignment constraints stated in equation (2) or (3), or the allocation restriction stated in equations (4) to (6). It is an illegal allocation which may have duplicate assignments and missing assignments of workstations, or does not satisfy the allocation restriction. An illegal allocation means that the given set of workstation position (\( PW_i \)) cannot be used. A new set must be generated. On the other hand, in the case where the given allocation does agree with the assignment constraints and the allocation restriction, it is legal, and it is able to continue to the next step.
Step 3: Conversion of given information parameters

Having obtained the legal allocation of workstations and assignment variable matrix \((P_y)\) from the above steps. The other given information parameters, such as the processing rate \(\mu_y\), failure rate \(F_i\), and recovery rate \(R_i\), which refer to workstation number \(i\), have to be converted to \(\mu_{S_j}\), \(F_{S_j}\), and \(R_{S_j}\), which refer to the allocation of workstation \(j\).

Conversion is necessary because further calculations in the model include some summations and equations which refer to the allocation number of workstation \(j\). As well, the small \(s\) in \(\mu_{S_j}\), \(F_{S_j}\), and \(R_{S_j}\) refers to stage. The processing rate \(\mu_y\) of workstation \(i\) is different for the workstation assigned to different positions \(j\). However, the failure rate \(F_i\) and recovery rate \(R_i\) are the same for a workstation assigned to different positions although each workstation has a different rate.

The following are the conversion equations.

- **conversion of processing rate**

  \[ \mu_{S_j} = \sum_{i=1}^{N} (\mu_{ij} \times P_{ij}) \quad \text{for } j = 1 \text{ to } N \]  

- **conversion of failure rate**

  \[ F_{S_j} = \sum_{i=1}^{N} (F_i \times P_{ij}) \quad \text{for } j = 1 \text{ to } N \]  

- **conversion of recovery rate**

  \[ R_{S_j} = \sum_{i=1}^{N} (R_i \times P_{ij}) \quad \text{for } j = 1 \text{ to } N \]
Equation (7) converts the processing rate of workstation \( i \) in position \( j \) (\( \mu_{ij} \)) to the processing rate of the workstation in stage \( j \) (\( \mu_j \)), by multiplying with the assignment variable \( P_j \). It may also be said that it converts the coefficient associated with the index for the number of workstations into the coefficient associated with the index for the number of stages. Equations (8) and (9) convert the failure rate and the recovery rate.

Other parameters are the inventory holding cost (\( HC_j \)) and storage cost (\( SC_j \)). These coefficients are associated with the index for the number of stages. Therefore, there is no need for conversion.

Step 4: Define the throughput rate

For an ideal JIT manufacturing system, the throughput rate should be equal to the demand rate. However, the workstations are unreliable, so the throughput rate must be larger than the demand rate to overcome the shortage of production during breakdowns of workstations. The processing rate of workstations relate to each other because the production line is a pull system. Consequently, the concept of steady state is employed, and applied to the whole system. When all the workstations and buffers in the production line reach a stable condition, the production is under steady state.

A workstation is also said to be steady when it has a constant throughput rate. Workstation states can change in several different ways. They may directly enter a steady state when the production system starts up, or after a period of transient state.
During the transient state period, the throughput rate of a workstation may be dependent upon that of the preceding or succeeding workstation, due to limited interstage buffer capacity or supply from the preceding workstation. Since the production line works in a JIT environment, all workstations and buffers will have the same throughput rate during the steady state. This was proved by Park and Steudel (1991).

Their research paper stated that, if the production line is in steady state, the processing rate of two workstations adjacent to a buffer will be equal to the one which has the lower rate. The equation as follows:

\[ \mu_i' = \mu_{i+1}' = \text{MIN} \{ \mu_i, \mu_{i+1} \} \]  \hspace{1cm} (10)

where \( \mu_i \) and \( \mu_{i+1} \) are the original processing rates for workstations allocated in position \( i \) and \( i + 1 \). In addition, \( \mu_i' \) and \( \mu_{i+1}' \) are their processing rates during steady state. Park and Steudel proved this equation by the following statement.

"If the workstation allocated in stage \( i + 1 \) has a lower processing rate than the workstation allocated in stage \( i \). The buffer in stage \( i + 1 \) will become full after entering into the steady state. Once the buffer is full, workstation \( i \) cannot transfer a completed item to the succeeding buffer \( i + 1 \). The workstation in position \( i \) is said to be blocked and the processing rate of the workstation allocated in position \( i \) depends on the processing rate of the workstation allocated in position \( i + 1 \). Therefore, the processing rate of the preceding workstation will be equal to the processing rate of the succeeding workstation. It will be the same as the case for the preceding workstation to have a lower processing rate (longer processing time), except the buffer will become empty."

Overall, the throughput rate of the steady state production line is determined in the following equation:

\[ \text{THR} = \text{MIN} \{ D_1, \mu s'_1, \mu s'_2, \mu s'_3, ..., \mu s'_n \} \]  \hspace{1cm} (11)

Here the throughput rate is the minimum effective processing rate \((\mu s'_j)\) of all workstations in the production system. The model also requires the consideration of the demand rate of the system; the throughput rate will be the demand rate if it is lower than the processing rate of all workstations.

The processing rate of the workstation in stage \(j\) \((\mu s'_j)\) does consider its failure and recovery rates. This is known as the effective processing rate. According to Gershwin (Gershwin 1985, 1987, and Choong and Gershwin 1987), the effective processing rate could be determined by the following equation:

\[ \mu s'_j = \mu s_j \times \frac{R_s_j}{F_s_j + R_s_j} \quad \text{for } j = 1 \text{ to } N \]  \hspace{1cm} (12)

The effective processing rate of the workstation in stage \(j\) \((\mu s'_j)\) is equal to the processing rate \((\mu s_j)\) multiplied by the recovery rate \((R_s_j)\), over the sum of the failure rate \((F_s_j)\) and the recovery rate \((R_s_j)\) of workstation \(j\). The failure rate \((F_s_j)\) is the rate of workstation failure over unit time. Also, the recovery rate \((R_s_j)\) is the rate of workstation recovery over unit time.

With the throughput rate determined, the next parameter required is the optimal number
of move kanbans for each interstage buffer. The number of kanbans is the maximum inventory level between two adjacent workstations. With the number of move kanbans, the inventory holding cost and shortage cost of WIP parts for each workstation can be determined. A major component in the total cost of production (\(TC\)) is the sum of these two costs (\(C_{2s}\)).

Wang and Wang (1990), proposed a model to find the optimal number of kanbans and these two costs for a similar production system in JIT environment. Their procedure for determining the optimum number of kanbans and the two expected costs between a pair of workstations is presented in the following steps.

**Step 5: Define the state space** \(E\) **for the production system**

Suppose the number of kanbans between a pair of workstations is \(K\). The production system can be described as a Markov process with \(NS\) number of states.

\[
NS = (K+1) \times 2 \times 2
\]

(13)

The above equation calculates the number of states for state space \(E\), with \(K\) number of kanbans. The number of states \(NS\) is equal to \(K + 1\) (possible states of kanban) multiplied by two of the possible status of the preceding workstation and two of the possible status of the succeeding workstation. For example, the number of kanbans between workstations \(A\) and \(B\) is equal to three. Then, the state space is \(E = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1), (2,0,0), (2,0,1), (2,1,0), (2,1,1)\).
(2,1,1), (3,0,0), (3,0,1), (3,1,0), (3,1,1), where the state components represent the following message: For state \((n, I1, I2)\), \(n (0 \leq n \leq K)\) is the number of fulfilled kanbans and \(I1, I2\) are the status indicators of workstations \(A\) and \(B\), respectively. The indicator \((I1, I2)\) is 1 when the workstation is up, and the indicator is 0 when the workstation is down (in the failure mode).

**Step 6: Define the transition matrix**

The transition matrix \(Q(k)\) for this Markov process with \(K = 0\), can be determined as follows.

For state \((0,0,0)\) all containers are empty and both workstations are down. In this situation, both workstations must be repaired. Therefore, the only two possible states that can be reached are \((0,0,1)\) or \((0,1,0)\). Similarly, other scenarios can be derived as shown in Table 3.1.

All possible changes for all states are shown in Table 3.1. However, the transition matrix \(Q\) does not need to consider all those states. It is an \(N\) by \(N\) matrix, which only considers the number of fulfilled kanbans. There are three possible situations for the transition matrix \(Q\) for this Markov process, according to the number of fulfilled kanbans.

1. For those states having empty containers (the first four rows in
Table 3.1), it assumes no production at the succeeding workstation. Either the preceding workstation manufactures parts as quickly as possible to fill the empty container, if the workstation is processing, or it is broken down, and the containers stay empty. In the queuing terminology, there is no departure rate but there are arrival and staying rates. The two possible states which can be reached are \((1, II, I2)\) and \((0, II, I2)\).

2. For states with full containers (the last four rows in Table 3.1), it assumes no production at the preceding workstation. Either the succeeding workstation consumes part as quickly as possible to free a space in the full containers if the workstation is processing, or it is broken down and the containers stay full. In queuing terminology, there is no arrival rate, but there are departure and staying rates. The two possible states which can be reached are \((K, II, I2)\) and \((K-1, II, I2)\).

3. For any other states of 1, 2, 3, ..., \(n\), ..., and \(K-1\) containers, (the middle four rows in Table 3.1), there are three possible state changes, which refer to the number of move kanbans. If the preceding workstation is processing, it manufactures parts to
increase the number of parts in containers. Otherwise, the succeeding workstation consumes parts to decrease the number of parts in containers, if it is processing, or the workstation/s (either one or both) is/are down and the containers stay at the same level. In queuing terminology, there are arrival, departure and staying rates. The three possible states which can be reached are \((n-1, I_1, I_2)\), \((n, I_1, I_2)\), and \((n+1, I_1, I_2)\).

<table>
<thead>
<tr>
<th>Current State</th>
<th>Next Possible States</th>
</tr>
</thead>
<tbody>
<tr>
<td>((0,0,0))</td>
<td>((0,0,1), (0,1,0))</td>
</tr>
<tr>
<td>((0,1,0))</td>
<td>((1,1,0), (0,1,1), (0,0,0))</td>
</tr>
<tr>
<td>((0,0,1))</td>
<td>((0,1,1), (0,0,0))</td>
</tr>
<tr>
<td>((0,1,1))</td>
<td>((1,1,1), (0,0,1), (0,1,0))</td>
</tr>
<tr>
<td>((n,0,0))</td>
<td>((n,0,1), (n,1,0))</td>
</tr>
<tr>
<td>((n,1,0))</td>
<td>((n+1,1,0), (n,1,1), (n,0,0))</td>
</tr>
<tr>
<td>((n,0,1))</td>
<td>((n,1,1), (n,0,0), (n-1,0,1))</td>
</tr>
<tr>
<td>((n,1,1))</td>
<td>((n+1,1,1), (n,0,1), (n,1,0), (n-1,1,1))</td>
</tr>
<tr>
<td>((K,0,0))</td>
<td>((K,1,0), (K,0,1))</td>
</tr>
<tr>
<td>((K,1,0))</td>
<td>((K,1,1), (K,1,0), (K-1,0,1))</td>
</tr>
<tr>
<td>((K,0,1))</td>
<td>((K,0,0), (K,1,1))</td>
</tr>
<tr>
<td>((K,1,1))</td>
<td>((K,0,1), (K,1,0), (K-1,1,1))</td>
</tr>
</tbody>
</table>

Table 3.1. Possible State Change.
The transition matrix \( Q \) can be computed from the above rules, and shown as such:

\[
Q_K = \begin{bmatrix}
p_4 & p_5 & 0 & \cdots & 0 & 0 \\
p_1 & p_2 & p_3 & 0 & \cdots & 0 \\
0 & p_1 & p_2 & p_3 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & p_1 & p_2 & p_3 & 0 \\
0 & \cdots & 0 & p_1 & p_2 & p_3 \\
0 & 0 & \cdots & 0 & p_6 & p_7 \\
\end{bmatrix}
\]  

(14)

In equation (14), \( p_1 \) and \( p_6 \) are the probabilities of going to the state with lower kanban level. Furthermore, \( p_2 \), \( p_4 \) and \( p_7 \) are the probabilities of staying in the current state. Finally, \( p_3 \) and \( p_5 \) are the probabilities of going to the state with higher kanban level.

All these probabilities can be determined by the following equations:

\[
P_1 = \frac{EF_{j-1}}{EF_{j-1} + IEF_{j-1} + EF_j + IEF_j}
\]

(15)

\[
P_2 = \frac{IEF_{j-1} + IEF_j}{EF_{j-1} + IEF_{j-1} + EF_j + IEF_j}
\]

(16)

\[
P_3 = \frac{EF_j}{EF_{j-1} + IEF_{j-1} + EF_j + IEF_j}
\]

(17)
\[ P_4 = \frac{p_2}{p_2 + p_3} \quad (18) \]

\[ P_5 = \frac{p_3}{p_2 + p_3} \quad (19) \]

\[ P_6 = \frac{p_1}{p_1 + p_2} \quad (20) \]

\[ P_7 = \frac{p_2}{p_1 + p_2} \quad (21) \]

Equations (18) and (19) determine \( p_4 \), and \( p_5 \) from \( p_1, p_2, \) and \( p_3 \). Equations (20) and (21) determine \( p_6 \), and \( p_7 \) from \( p_1, p_2, \) and \( p_3 \). Equations (15), (16), and (17) calculate \( p_1, p_2, \) and \( p_3 \). However, the effective processing rate \( EF_j \), and the ineffective processing rate \( IEF_j \), have to be determined first. According to the research by Gershwin (Gershwin 1985, 1987, Choong and Gershwin 1987), both rates could be determined by the following equations:

\[ EF_j = \mu s_j' = \mu s_j \times \frac{R_s}{R_s + F_s} \quad \text{for} \ j = 1 \ \text{to} \ N \quad (22) \]
\[ IEF_j = \mu_j \times \left( 1 - \frac{R_{s_j}}{R_{s_j} + F_{s_j}} \right) \quad \text{for } j = 1 \text{ to } N \] (23)

Step 7: Determine the limiting distribution

The limiting distribution for the Markov process can be determined in the following way:

1. Determine the unique solution of \( \nu = \nu Q_k \), where \( \nu = (\nu_0, \nu_1, \nu_2, \ldots, \nu_{k-2}, \nu_{k-1}, \nu_k) \).

2. Determine the exponential distribution for staying in each state \( (\lambda) \). The expected number of visits to other states \( j \) between two visits to a certain state \( l \) is \( \nu(j)/\nu(l) \) and the expected time of staying in state \( l \) is \( 1/\lambda(l) \), where \( \lambda = (\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{k-2}, \lambda_{k-1}, \lambda_k) \), which is the parameter of exponential distribution for staying in each state.

3. Finally, the expected time spent in state \( l \), which is the limiting distribution, can be determined by the following equation:

\[ \pi(l) = \frac{\nu(l)}{\sum_{j=0}^{K} \frac{\nu(j)}{\lambda(j)}} \quad \text{for } l = 1 \text{ to } K \] (24)
Step 8: Determine the cost factor for each state

Suppose that:

1. The shortage cost of WIP items for workstation \( j + 1 \) is \( SC_j \) per unit time, which is different for different positions, but independent of workstations.

2. The inventory holding cost for one kanban of items at workstation \( j \) is \( HC_j \) per unit time per kanban. It is also different for different positions, and independent of workstations.

The cost factor for each state can be determined as follows.

1. For state \( l \) with state vector \((0, II, I2)\), the cost factor \( C_l \) is \( SC_j \).

2. For state \( l \) with state vector \((K, II, I2)\), the cost factor \( C_l \) is \( HC_j \times K \) (assuming there is no idle cost for the preceding workstation \( j \)).

3. For state \( l \) with state vector \((l, II, I2)\), where \( 0 < l < K \), the cost factor \( C_l \) is \( HC_j \times l \).

Step 9: Determine the expected cost (of inventory holding and WIP shortage) for the production system with \( K \) kanbans

The expected cost for inventory holding and WIP shortage \((C2)\) can be determined by the following equation, according to the model developed by Wang and Wang (1990).
This expected cost relates highly to the number of kanbans used in the production system.

\[ C_{2s_j} = \sum_{l=0}^{K} \pi_K(l) \times C_i \]  

(25)

Step 10: Determine the optimal number of kanbans \( (K) \)

Repeat steps (5) to (9) for different values of \( K = 1, 2, 3, \ldots \), until there is no further improvement in the expected cost (lower). The value of \( K \) with the lowest expected cost will be the optimal number of kanbans \( K_{s_j} \) and the expected cost will be \( C_{2s_j} \) for stage \( j \). Repeat steps (5) through (9) to determine the optimal number of kanbans and expected cost for the next interstage buffer until the last buffer.

Step 11: Determine the starving time of the system for the breakdown of a workstation

A broken down workstation in the production line will cause starvation in its succeeding workstation if there are not enough parts in the buffer to supply it during the recovery. If the recovery time is too long (low recovery rate), all succeeding buffers become empty, and all succeeding workstations are starving. Consequently, the production system will have no more supply of the finished product; the production line is starving and cannot meet the demand. Thus, shortage cost of finished products (\( C_{3j} \)) is applied to the production system. To determine this shortage cost of finished products, the mean starving time of the system first needs to be obtained. This step defines the starving time.
(\textit{ST}_{ sj}) of the system for the breakdown of workstation \( j \).

Park and Steudel's (1991) proof that the inventory level of the buffer in a JIT production line is either empty or full during steady state. This proof has already been shown in step 3. The following equation is developed to determine the inventory level \( I_{ sj} \) for the interstage buffer in stage \( j \):

\[
I_{ sj} = \begin{cases} 
K_{ sj} & \text{for } \mu_{s_{j-1}} > \mu_{s_{j}} \\
0 & \text{otherwise}
\end{cases} \quad \text{for } j = 2 \text{ to } N \tag{26}
\]

The optimal buffer size (number of kanbans) \( K_{ sj} \) is obtained in step 10. The starving time \( (\text{ST}_{ sj}) \) of the system for the breakdown of workstation in stage \( j \) can be determined by the following equation.

\[
\text{ST}_{ sj} = \frac{1}{R_{ sj}} + \sum_{k_{ j}}^{N} \frac{1}{\mu_{ s_{ k_{ j}}}} - \sum_{k_{ j-1}}^{N} [(I_{ sj} + 1) \times \frac{1}{\mu_{ s_{ k_{ j-1}}}}]
\]

\text{for } j = 1 \text{ to } N - 1 \tag{27}

The starving time \( \text{ST}_{ sj} \) is calculated as the re-supply time of the production line after the workstation breakdown minus the mean time to starving. The line's re-supply time after the breakdown of the workstation in stage \( j \) is its recovery time \( 1/R_{ sj} \) (inverse of the recovery rate) plus the summation of the processing time (inverse of the processing rate) of workstation in stage \( j \) and all succeeding workstations. This is the time required to produce a finished item from the breakdown workstation to the last workstation. The
mean time to starving is the time for which the succeeding workstation is operable before
starving occurs due to the lack of supply from the breakdown workstation. It is the
summation of the inventory in all succeeding buffers plus the one item processing in each
succeeding workstation located after the breakdown one. As well, it multiplies by the
processing time \( \frac{1}{\mu S_i''} \) of the succeeding workstations during steady state, which is
calculated as below.

\[
\mu S_j'' = \text{MIN} \{ \mu S_j, \mu S_{j-1}, \ldots, \mu S_N \} \quad \text{for } j = 1 \text{ to } N
\]  

(28)

The value of starving time \( STS_j \) obtained in here may be negative, if the re-supply time
of the production line for the breakdown of workstation \( j \) is less than the mean time to
starving. In this case, there is no starving. Therefore, the starving time \( STS_j' \) is equal
to zero, as mentioned in equation (29).

\[
STS_j' = \text{MAX} \{ STS_j, 0 \} \quad \text{for } j = 1 \text{ to } N - 1
\]  

(29)

For the breakdown of the workstation allocated in last position, there is no mean time
to starving, as there is no succeeding workstation and buffer. The system starves right
after the last workstation breaks down. This situation continues until the workstation
recovers and processes one item. Hence, the starving time \( STS_N' \) is the breakdown time
\( (1/Rs_N) \) plus the processing time \( (1/\mu S_N) \) of the workstation allocated in the last position
\( N \). This is shown in the following equation.
\[ STS'_{N} = \frac{1}{R_{N}} + \frac{1}{\mu_{N}} \] (30)

Since the breakdown of the last workstation has no mean time to starving, there will not be a negative value for the starving time \( STS'_{N} \). There is no need to check by using equation (29).

Step 12: Determine the mean starving time for the production system

Finally, with the positive value of starving time \( STS'_{N} \), the mean starving time of the production system can be determined as such:

\[ MST = \sum_{j=1}^{N} \left[ \left( 1 - \frac{R_{j}}{R_{j} + F_{j}} \right) \times STS'_{j} \right] \] (31)

In equation (31), the probability of workstations' breakdown is one minus the reliability (the recovery rate over the sum of the recovery and failure rates). This is proved by Gershwin (Gershwin 1985, 1987, Choong and Gershwin 1987). Then it is multiplied with the starving time \( STS'_{j} \) and the summation generates the mean starving time \( MST \).

Step 13: Determine the expected total cost of production per item

With all required information given in the above steps, it is possible to determine the expected total cost of production (TC) for each finished product. The variables given are the throughput rate (THR) defined in step 4 by equation (12), the expected cost for inventory holding and WIP shortage (C2s_j) determined in step 10 for all \( N-1 \) interstage
buffers, and the mean starving time \((MST)\) obtained in step 12 by equation (31). These three values will apply to the following equation to determine the expected total cost of production \((TC)\).

\[
TC = C1\left(\frac{1}{THR}\right) + \frac{1}{D} \sum_{j=2}^{N} C2s_j + C3(MST) \tag{32}
\]

Equation (32) can be divided into three portions. The first portion calculates the operation cost to produce one item. The operation cost \(C1\) is the sum of direct labour cost, machine cost, maintenance cost, overhead cost, and all other operational costs which relate to the allocation of workstations. This operation cost \(C1\) is a given variable with the unit of dollars per unit time. It multiplies by the throughput time \((1/THR\), the inverse of the throughput rate \(THR\)\) of the system in steady state, to determine the operation cost per item.

The second portion calculates the inventory holding cost plus the shortage cost of WIP parts for the production system per each item it produces. It is the sum of the inventory holding and shortage costs of WIP parts in each interstage buffer per unit time \((C2s_j)\) for all \(N-1\) buffers. After that, it is divided by the demand rate per unit time \((D)\). Both the inventory holding cost and the shortage cost of WIP parts relate highly to buffer capacity. The buffer capacity is controlled by the number of move kanbans.

The third portion of equation (32) obtains the cost of shortage in finished products. This
shortage cost of finished products $C_3$ has the unit of dollars per unit time. The shortage cost $C_3$ multiplies by the mean starving time $MST$ to generate the shortage cost of finished products per each item produced in equation (31).

With this equation, the expected total cost of production per item ($TC$) for each different workstations arrangement is able to be determined. It is also possible to verify which workstations arrangement has a better performance, by comparing their expected total cost of production per item ($TC$).

In the following chapter, the model developed here will be applied to an numerical example of a production line with five unreliable workstations operating in a JIT environment.
CHAPTER 4

APPLICATION OF THE MATHEMATICAL MODEL

4.1 Numerical Examples

In this chapter, a numerical example is used to illustrate the solution procedures presented in Chapter 3. The mathematical model is applied to an example of a five workstations production line ($N = 5$). It is assumed that the production line starts at empty and all workstations are in idle. The allocation of workstations is $PW = (1, 2, 3, 4, 5)$; workstation 1 will be allocated to stage 1, workstation 2 will be allocated to stage 2, etc. The processing rates ($\mu_y$) for the five workstations in each position are shown in Table 4.1. The unit for the processing rate is item per hour.

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Processing Rate at Stage 1</th>
<th>Processing Rate at Stage 2</th>
<th>Processing Rate at Stage 3</th>
<th>Processing Rate at Stage 4</th>
<th>Processing Rate at Stage 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>20</td>
<td>35</td>
<td>45</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>40</td>
<td>30</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>24</td>
<td>30</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>45</td>
<td>30</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>55</td>
<td>35</td>
<td>25</td>
<td>30</td>
<td>45</td>
</tr>
</tbody>
</table>
The failure rate \( (F_i) \) and recovery rate \( (R_i) \) are shown in Table 4.2. Their units are also item per hour.

<table>
<thead>
<tr>
<th>Workstation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Rate</td>
<td>0.25</td>
<td>0.5</td>
<td>0.25</td>
<td>0.1</td>
<td>0.8</td>
</tr>
<tr>
<td>Recovery Rate</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4.2. Failure Rates and Recovery Rates for Workstations.

There are also inventory holding costs \( HC_{S_j} \) and WIP shortage costs \( SC_{S_j} \), related to the work stage. They are presented in Table 4.3.

<table>
<thead>
<tr>
<th>Stage ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( HC_{S_j} )</td>
<td>--</td>
<td>20</td>
<td>20</td>
<td>35</td>
<td>50</td>
</tr>
<tr>
<td>( SC_{S_j} )</td>
<td>--</td>
<td>200</td>
<td>110</td>
<td>155</td>
<td>180</td>
</tr>
</tbody>
</table>

Table 4.3. Inventory Holding Cost and WIP Shortage Cost for Work Stages.

Other given variables are shown below. There is no restriction in the allocation of workstations.

- \( C1 = $240 / \text{hr} \)
- \( C3 = $50 / \text{hr} \)
- \( D = 20 \)
4.2 Application of the Mathematical Model

With the given information, the numerical example is applied to the mathematical model. The procedure to determine the expected total cost of production ($TC$) is presented in the following steps.

Step 1: Generate the assignment variable matrix

After allocating workstations (shown above), this step determines the assignment variable of workstations ($P_{ij}$), by applying equation (1).

$$
P_{ij} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Step 2: Check for illegal allocation

The assignment variable matrix of this allocation of workstations does agree with equation (2) and (3); the allocation is legal. This example does not have any allocation restrictions; there is no need to check by using equations (4), (5), and (6).

Step 3: Convert the given information parameters

Convert the processing rate $\mu_i$ with equation (7). The result is shown in the following.
\[ \mu_{s,j} = (50,40,30,25,45) \quad \text{for } j = 1 \text{ to } 5 \]

Convert failure and recovery rates with equations (8) and (9).

\[ \delta_{s,j} = (0.25,0.5,0.25,0.1,0.8) \quad \text{for } j = 1 \text{ to } 5 \]

\[ R_{s,j} = (4.6,4.2,10) \quad \text{for } j = 1 \text{ to } 5 \]

**Step 4: Define the throughput rate**

Calculate the throughput rate of the production system using equation (11), after determining the effective processing rate of all workstations.

\[ \mu_{s,j}' = (47.06,36.92,28.24,23.81,41.67) \quad \text{for } j = 1 \text{ to } 5 \]

\[ THR = \text{MIN} \{ 20,47.06,36.92,28.24,23.81,41.67 \} = 20 \]

**Step 5: Define the state space \( E \) for the production system**

The next procedure determines the buffer size and the related cost, the expected cost for inventory holding and WIP shortage (C2). These are determined buffer by buffer. For the first buffer in stage 2 \( K_{s2} \), it is assumed equal 1. The number of states \( NS = 8 \), which was defined by equation (13). The state space is \( E = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1) \).
Step 6: Define the transition matrix

Before calculating the transition matrix, the probabilities $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, and $p_7$ need to be defined. The effective processing rate $EF_j$ and the ineffective processing rate $IEF_j$ are needed for the computation of the probabilities. They are determined by equations (22) and (23) as illustrated.

$$EF_j = \mu s_j' = (47.06, 36.92, 28.24, 23.81, 41.67)$$

for $j = 1$ to $5$

$$IEF_j = (2.94, 3.08, 1.76, 1.19, 3.33)$$

for $j = 1$ to $5$

For the first buffer in stage 2 $Ks_2 = 1$, which is between the workstations in stages 1 and 2. The probabilities are defined using equations (15) through (21).

$$p_1 = 0.52, p_2 = 0.07, p_3 = 0.41, p_4 = 0.14,$$

$$p_5 = 0.86, p_6 = 0.89, p_7 = 0.11$$

With the probabilities established, the transition matrix $Q_i$ for the buffer in stage 2 may be determined.

$$Q_1 = \begin{bmatrix} p_4 & p_5 \\ p_6 & p_7 \end{bmatrix} = \begin{bmatrix} 0.14 & 0.86 \\ 0.89 & 0.11 \end{bmatrix}$$
Step 7: Determine the limiting distribution

Solving $\nu = \nu - Q_k$, the value of $\nu_j$ is obtained.

$$\nu(0) = 0.51, \quad \nu(1) = 0.49$$

The exponential distribution for staying in each state is

$$\lambda(0) = 50, \quad \lambda(1) = 40$$

Finally, from equation (24), the limiting distribution is determined.

$$\pi(0) = 0.45, \quad \pi(1) = 0.55$$

Step 8: Determine the cost factor for each state

With the inventory holding cost $HCs_j$ and WIP shortage cost $SCs_j$ of Table 4.3, the cost factor for each is calculated as following.

$$C_0 = SCs_2 = 200$$

$$C_1 = HCs_2 \times K = 20 \times 1 = 20$$

Step 9: Determine the expected cost (of inventory holding and WIP shortage) for the production system with $K$ kanbans

From equation (25), the expected cost $C2s_2$ could be determined, with the result as
follows.

\[ C_{2s_2} = \sum_{j=0}^{1} \pi_K(j) \times C_j = \pi_K(0) \times C_0 + \pi_K(1) \times C_1 \]

\[ C_{2s_2} = 0.45 \times 200 + 0.55 \times 20 = 101.36 \]

Step 10: Determine the optimal number of kanbans \((K)\)

Repeat steps (5) to (9) for different values of \(K = 1, 2, 3, \ldots\), until there is no further improvement in expected cost. The expected cost for different values of \(K\) is determined and showed in Table 4.4.

<table>
<thead>
<tr>
<th>(Ks_2)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C_{2s_2})</td>
<td>101.36</td>
<td>89.47</td>
<td>86.39</td>
<td>86.69</td>
</tr>
</tbody>
</table>

From Table 4.4, the number of kanbans with the lowest expected cost is 3. Therefore, the optimal number of kanbans \(Ks_j\) for the interstage buffer in stage 2 \((j = 2)\) is three. The expected cost \(C_{2s_j}\) for this buffer is $86.39 per hour.

It is now necessary to repeat the procedure for the rest of the buffers. The results are presented in Table 4.5.
Table 4.5. Number of Kanbans and Expected Cost for Stage \( j \).

<table>
<thead>
<tr>
<th>Stage ( j )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K_s_j )</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( C_{2s_j} )</td>
<td>86.39</td>
<td>57.70</td>
<td>87.66</td>
<td>110.91</td>
</tr>
</tbody>
</table>

Step 11: Determine the starving time of the system for the breakdown of a workstation.

The first variable to determine is the inventory level \( I_s_j \). This is accomplished with equation (26), as shown.

\[
I_s_j = (3, 2, 2, 0) \quad \text{for } j = 2 \text{ to } 5
\]

Then, find the processing rate of the succeeding workstation during steady state \( \mu s_j'' \) using equation (23).

\[
\mu s_j'' = (25, 25, 25, 45) \quad \text{for } j = 2 \text{ to } 5
\]

From equation (27), the starving time \( STs_j \) can be determined.

\[
STs_j = \frac{1}{Rs_j} + \sum_{l=1}^{5} \frac{1}{\mu s_l} - \sum_{l=2}^{5} \frac{(I_s_j) \times \frac{1}{\mu s_l''}}{\mu s_l''}
\]
After that, the starving time $STs_j$ for workstations in stages 1 to 4 are obtained as presented in Table 4.6.

<table>
<thead>
<tr>
<th>Stage $j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$STs_j$</td>
<td>-0.03</td>
<td>0.025</td>
<td>0.203</td>
<td>0.54</td>
</tr>
</tbody>
</table>

Since the starving time $STs_j$ should have positive values, they have to be examined by equation (29). The approved starving time $STs_j^*$ are shown in the following table.

<table>
<thead>
<tr>
<th>Stage $j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$STs_j^*$</td>
<td>0</td>
<td>0.025</td>
<td>0.203</td>
<td>0.54</td>
</tr>
</tbody>
</table>

The next procedure is to determine the starving time $STs_j^*$ for the workstation at the last stage ($j = 5$). This is done by using equation (30).

$$STs_5^* = \frac{1}{RS_5} + \frac{1}{\mu s_5} = \frac{1}{10} + \frac{1}{45} = 0.12$$
Step 12: Determine the mean starving time for the production system

With the starving time $ST_s'$ determined, the mean starving time MST can be calculated by equation (31).

$$MST = \sum_{j=1}^{5} \left[ \left( 1 - \frac{R_{s_j}}{R_{s_j} + F_{s_j}} \right) \times ST_{s_j}' \right] = 0.0487$$

Step 13: Determine the expected total cost of production per item

Finally, with all required data gathered, it is possible to determine the expected total cost of production ($TC$) for each finished product. This is done using equation (32).

$$TC = 240 \left( \frac{1}{20} \right) + \frac{1}{20} (86.39 + 57.7 + 87.66 + 110.91) + 50 (0.0487)$$

$$TC = 31.57$$

As a result, the expected total cost $TC$ for this allocation of workstations, shown in Table 4.8, is $31.57 per each item produced. For a second case in the same production line, a different allocation of workstations is stated in Table 4.8.

For the second allocation of workstations, the expected total cost of production ($TC$) determined by the mathematical model is $31.42 per item produced. This means that the second allocation is better than that of case 1, since its expected total cost of production is lower than the first one's.
Table 4.8. Allocation of Workstations for Case 1 and Case 2.

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Case 1 Allocation</th>
<th>Case 2 Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

The objective of this mathematical model is to determine the total production cost of one item for a specific position allocation of workstations. It also defines the best position allocation of workstations for a production line system, which is an overall objective of this research.
CHAPTER 5

GENETIC ALGORITHM

5.1 Mechanics of Genetic Algorithm

Before applying the genetic search algorithm to the model developed in the previous chapter, a brief discussion of the mechanics of the simple genetic algorithm is presented.

A simple genetic algorithm is mostly composed of three simple operators:

- reproduction
- crossover
- mutation.

Reproduction copies individual strings according to their objective function value. The string(s) of high value will be reproduced. The copied string replaces the one with the lowest objective function value. With the newly reproduced strings as the mating pool, pairs of strings are mated at random. This is known as crossover. Each pair undergoing crossover will swap partially at a random position. The following is an example of crossover, with a pair of 5-bit strings with binary numbers A1 and A2, known as initial strings (parent):

\[
\begin{align*}
A1 &= 0 \ 1 \ 0 \ 1 \ 1 \\
A2 &= 1 \ 0 \ 0 \ 0 \ 1 \\
A1' &= 0 \ 1 \ 0 \ 0 \ 1 \\
A2' &= 1 \ 0 \ 0 \ 1 \ 1 
\end{align*}
\]
First, a position allocation value is randomly generated to separate the string in two portions (the number generated is 2, meaning crossover at the second digit, indicated by \}). The front portion of string A1 is then cross mated with the rear portion of string A2. The same operation occurs for string A2. A new pair of strings, A1' and A2', are generated, and known as children or offspring.

Both reproduction and crossover efficiently build new solutions from the best partial solutions of previous trials. Although reproduction and crossover effectively generate and recombine improved offspring, they may become overzealous and lose some potentially useful genetic material. Mutation is needed to protect against such an irrecoverable loss. An example of mutation is shown below. A position value of 2 is randomly generated in the 5-bit string of binary A1. The value in that position changes from a 0 to a 1, or from a 1 to a 0, to generate a new string A1'.

\[
\begin{align*}
A1 &= 0 \ 1 \ 1 \ 0 \ 0 \\
A1' &= 0 \ 1 \ 1 \ 1 \ 0
\end{align*}
\]

In simple genetic algorithms, mutation is the occasional random alteration of the value of a string position. The probability of mutation is very small, about one per thousand bit.

5.2 Model Application

In the mathematical model, the decision variable requiring determination is the allocation
order of workstations. The order is a string with the value 1 to \( N \) (number of workstations). Inside the allocation string, the values cannot be repeated, but they could be in any order.

The allocation string indicates the allocation order \((PW_i)\) of the workstations. An example allocation string \((A)\), with 9 workstations, is shown in the following table.

<table>
<thead>
<tr>
<th>String A</th>
<th>PW₁</th>
<th>PW₂</th>
<th>PW₃</th>
<th>PW₄</th>
<th>PW₅</th>
<th>PW₆</th>
<th>PW₇</th>
<th>PW₈</th>
<th>PW₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

The position of workstation 1 \((PW₁)\) is 5 meaning that it is allocated in the fifth position.

The application of the genetic algorithm to the workstation allocation problem could be divided into seven steps:

1. Input the number of workstations \(N\), the processing rate for each workstation in different positions, the failure rate and recovery rate for each workstation, and other required data.

2. Randomly generate a set of legal strings (allocation strings).

3. Evaluate the performance of each string with the mathematical model developed in Chapter 3.


5. Check that each new string is legal, replace an illegal string by generating a new
legal string.

6. Re-evaluate each new string using the mathematical model.

7. Repeat steps 4, 5, and 6 until no more improvement is achieved.

After input of required data, a set of legal strings is randomly generated. This is the allocation string, which is the allocation of workstation \(PW_i\). The establishment of the set of legal allocation strings is similar to the Travelling Salesman Problem (TSP). It is a salesman travels through several cities in which legal sequences must be preserved. No city may be missed or repeated. Similarly, in the allocation problem, a workstation cannot be allocated to more than one position, and no workstation can be missed. All the values in a legal allocation string must be from 1 to \(N\), with no missing and duplication of workstations.

Each string must also be checked against the allocation restrictions in step 2 of the mathematical model in Chapter 3. Any illegal string must be replaced by a new string. The new string must be further checked for legality.

The number of strings in a set could be any number. For this problem, seven allocation strings are generated in each set. These initial strings can be generated randomly, or by some specific configurations such as the bowl phenomenon. In this research, four of the initial strings are generated by several specific configurations:

- increasing order of workstations reliability \(r_i\),

66
• decreasing order of workstations reliability \( r_i \),

• order in bowl phenomenon with lowest workstations reliability \( r_i \) in the centre.

• order in bowl phenomenon with highest workstations reliability \( r_i \) in the centre.

The reliability of each workstation could be determined by the following equation.

\[
    r_i = \frac{R_i}{R_i + F_i}
\]  

(33)

The rest of the initial strings are randomly generated.

The next step is to evaluate each string using the mathematical model. After that, genetic algorithm operation is performed. The best allocation string is preserved for later comparison. The two worst strings are replaced by the two best strings, in reproduction. The set of six allocation strings, excluding the best one, is randomly mated in crossover. Furthermore, mutation is performed in the probability of 1 per thousand of activities.

In simple genetic algorithm operation, crossover will generate illegal strings for this problem. Therefore, another crossover technique, Partially Mapped Crossover, is used. Partially Mapped Crossover (PMX) allows no missing and no duplication of values in the string. It was developed by Goldberg and Lingle in 1985 for the Travelling Salesman Problem. An example of applying the Partially Mapped Crossover to the allocation
string is illustrated below.

A pair of allocation strings, A and B, with the allocation order of 9 workstations to 9 positions are the initial strings. Two position values are randomly generated as the crossover points for the strings. The portions of strings between the two values are interchanged to form strings A' and B'. However, strings A' and B' are a pair of illegal strings, both having duplication (underlined) and missing allocation. The problem is corrected by replacing the duplicated workstations outside the crossover range by the missing workstations. The process modifies the illegal strings A' and B' into a legal pair, A'' and B''.

\[
\begin{align*}
\text{String A} & : 5 \ 6 \ | 2 \ 9 \ 3 \ 4 | \ 7 \ 1 \ 8 \\
\text{String B} & : 6 \ 2 \ | 8 \ 3 \ 7 \ 5 | \ 9 \ 4 \ 1 \\
\text{String A'} & : 5 \ 6 \ | 8 \ 3 \ 7 \ 5 | \ 7 \ 1 \ 8 \\
\text{String B'} & : 6 \ 2 \ | 2 \ 9 \ 3 \ 4 | \ 2 \ 4 \ 1 \\
\text{String A''} & : 4 \ 6 \ | 8 \ 3 \ 7 \ 5 | \ 9 \ 1 \ 2 \\
\text{String B''} & : 6 \ 8 \ | 2 \ 9 \ 3 \ 4 | \ 7 \ 5 \ 1
\end{align*}
\]

Mutation simply exchanges the position of two workstations with the probability of 1 per thousand. Here is an example of string A with 9 workstations:

\[
\begin{align*}
\text{String A} & : 5 \ 6 \ 2 \ 9 \ 3 \ 4 \ 7 \ 1 \ 8 \\
\text{String A'} & : 5 \ 6 \ 7 \ 9 \ 3 \ 4 \ 2 \ 1 \ 8
\end{align*}
\]

Two position values that are not the same are randomly generated. Then the allocation
of a workstation in that position (underlined) is interchanged to form a new mutated string $A^\prime$.

Finally, the new set of six strings is produced with the best string of the last trial, to form a set of seven strings. The newly generated set of strings is legal, without duplication or missing allocation. However, it needs to be checked by the allocation restrictions and re-evaluated by the mathematical model. After the evaluation, the above search is performed again, to create another new set of strings.

Steps 4, 5, and 6 are repeated continuously, until no string with better performance function is available. This termination process is controlled by a termination factor ($TF$). The termination factor is the number of trials in the repetition of steps 4, 5, and 6 performed without any improvement. This termination factor is directly proportional to the number of workstations in the production system. The equation for calculating the termination factor is shown in the following.

$$TF = \alpha N$$ (34)

$\alpha$ is the termination constant, which will be determined in the following section.

5.3 Application of Genetic Algorithm to the Numerical Example

Genetic algorithm is applied to the numerical example developed in the last chapter, and the required data is given in Section 4.1. A computer program in FORTRAN language,
GA.FOR (see Appendix A), is developed to solve the problem of workstations allocation by the genetic algorithm. The value of the termination factor ($TF$) used for this numerical example is 15; the search will terminate after 15 more trials without any improvement in the objective function (expected total cost of production $TC$). The initial set of strings generated by the genetic algorithm is shown in the following table.

Table 5.2. Initial Set of Strings for the Numerical Example.

<table>
<thead>
<tr>
<th>String</th>
<th>Allocation Strings ($PW_i$)</th>
<th>Total Cost ($TC_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2,5,1,3,4)</td>
<td>34.01</td>
</tr>
<tr>
<td>2</td>
<td>(4,3,1,5,2)</td>
<td>31.93</td>
</tr>
<tr>
<td>3</td>
<td>(3,5,2,1,4)</td>
<td>32.69</td>
</tr>
<tr>
<td>4</td>
<td>(5,3,4,1,2)</td>
<td>32.42</td>
</tr>
<tr>
<td>5</td>
<td>(1,2,3,4,5)</td>
<td>31.57</td>
</tr>
<tr>
<td>6</td>
<td>(4,5,1,3,2)</td>
<td>31.99</td>
</tr>
<tr>
<td>7</td>
<td>(1,4,3,2,5)</td>
<td>31.73</td>
</tr>
</tbody>
</table>

The average expected total cost of production for the initial set of strings is $32.33 per item. The strings' set of the second trial is shown in Table 5.3. The average expected total cost of production for the second trial is $31.95 per item, an improvement compared with the initial set. The optimal (near-optimal) allocation of workstations is obtained in the fifth trial, with an expected total cost of production of $31.02 per item. The search is terminated at trial 20, with an average expected total cost of production of $31.02 per item. The optimal, or near-optimal, allocation of workstations determined
by the genetic algorithm for this numerical example is illustrated in Table 5.4. The optimal, or near-optimal, number of kanbans for this allocation is found in Table 5.5.

Table 5.3. Second Set of Strings for the Numerical Example.

<table>
<thead>
<tr>
<th>String</th>
<th>Allocation Strings ($PW_i$)</th>
<th>Total Cost ($TC$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(5,3,1,2,4)</td>
<td>31.76</td>
</tr>
<tr>
<td>2</td>
<td>(4,5,2,3,1)</td>
<td>32.13</td>
</tr>
<tr>
<td>3</td>
<td>(1,2,3,4,5)</td>
<td>31.57</td>
</tr>
<tr>
<td>4</td>
<td>(3,2,4,1,5)</td>
<td>32.16</td>
</tr>
<tr>
<td>5</td>
<td>(1,4,3,2,5)</td>
<td>31.73</td>
</tr>
<tr>
<td>6</td>
<td>(2,3,4,5,1)</td>
<td>32.40</td>
</tr>
<tr>
<td>7</td>
<td>(1,2,3,4,5)</td>
<td>31.57</td>
</tr>
</tbody>
</table>

Table 5.4. Solution Allocation of Workstation for the Numerical Example.

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Allocation ($PW_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 5.5. Number of Kanbans for this Allocation of Workstations.

<table>
<thead>
<tr>
<th>Stage ((j))</th>
<th>Number of Kanbans ((K_{ij}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

5.4 Discussion of Result for the Numerical Example of Allocating 5 Workstations

For this numerical problem of allocating 5 workstations, the number of possible allocations is the factorial of the number of workstations \((N)\).

\[
N! = 5! = 120
\]

The expected total cost of production for all of these allocations is determined by FORTRAN program TC.FOR (Appendix C). The optimal allocation is the same as the solution obtained by the genetic algorithm of Table 5.4. Therefore, the feasible solution acquired by the genetic algorithm is the optimal solution for this numerical example. However, the feasible solution obtained is not guaranteed to be the optimal solution for other example.

There is a problem of duplication strings in a set for the application of genetic algorithm in the previous section. In the final trial (20), all the allocation strings are the same. This is the optimal allocation, caused by replacing the worst string with the best one.
during the reproduction process. The best string reproduces until all the strings are the same. After that, genetic algorithm generates a new set of strings those which are all the same, producing a new set which is also the same. This will trap the search in a local minimum. The problem of duplication strings in a set could be solved by increasing the number of strings in each set. Another test is performed, with an increase of strings from 7 to 9 in a set. However, the same problem occurs in the final trial, and the number of strings must be further increased. Finally, the problem is solved with a set of 11 strings. The final set of strings, found in the 19th trial, is shown in Table 5.6.

Table 5.6. The Final Set of Strings for the Test with 11 Strings in a Set.

<table>
<thead>
<tr>
<th>String</th>
<th>Workstations Allocation</th>
<th>Expected Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5, 1, 3, 4, 2</td>
<td>31.42</td>
</tr>
<tr>
<td>2</td>
<td>1, 3, 5, 2, 4</td>
<td>31.80</td>
</tr>
<tr>
<td>3</td>
<td>4, 3, 1, 2, 5</td>
<td>32.48</td>
</tr>
<tr>
<td>4</td>
<td>3, 5, 1, 4, 2</td>
<td>32.49</td>
</tr>
<tr>
<td>5</td>
<td>5, 3, 1, 2, 4</td>
<td>31.76</td>
</tr>
<tr>
<td>6</td>
<td>2, 3, 1, 4, 5</td>
<td>33.17</td>
</tr>
<tr>
<td>7</td>
<td>5, 3, 1, 4, 2</td>
<td>31.02</td>
</tr>
<tr>
<td>8</td>
<td>5, 1, 4, 3, 2</td>
<td>32.56</td>
</tr>
<tr>
<td>9</td>
<td>5, 3, 1, 2, 4</td>
<td>31.76</td>
</tr>
<tr>
<td>10</td>
<td>5, 3, 1, 4, 2</td>
<td>31.02</td>
</tr>
<tr>
<td>11</td>
<td>5, 3, 1, 4, 2</td>
<td>31.02</td>
</tr>
</tbody>
</table>
Table 5.7. Average Expected Cost and Feasible Solution for Each Trial.

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>Average Cost of Production</th>
<th>Feasible Solution</th>
<th>Expected Cost of Production for Feasible Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32.5805</td>
<td>4, 3, 1, 5, 2</td>
<td>31.9340</td>
</tr>
<tr>
<td>2</td>
<td>32.2368</td>
<td>4, 1, 5, 3, 2</td>
<td>31.1367</td>
</tr>
<tr>
<td>3</td>
<td>31.9825</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>31.6191</td>
<td>4, 3, 5, 1, 2</td>
<td>31.0740</td>
</tr>
<tr>
<td>5</td>
<td>31.4709</td>
<td>5, 3, 1, 4, 2</td>
<td>31.0201</td>
</tr>
<tr>
<td>6</td>
<td>31.6237</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>32.6585</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>32.1232</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>31.7834</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>31.8073</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>32.1251</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>32.2120</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>31.9361</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>31.8900</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>31.9367</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>31.6081</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>31.7413</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>31.4614</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>31.8636</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

In this final set of strings, the average expected total cost of production is $31.86. The optimal, or near-optimal, allocation of workstations is the same as the first test with 7
strings in a set. Three of the strings are the same in this final set, which is the optimal solution. This situation is acceptable because the duplication of strings is generated by the process of reproduction.

The optimal workstation allocation is obtained in trial 5, and the search is terminated after 15 trials (termination factor) without improvement, which is at trial 19. The average expected cost of production and the feasible solution for each trial are listed in Table 5.7. During the processing of genetic algorithm, new feasible solutions are obtained in trials 1, 2, 4, and 5. A graph of trial versus average expected cost of production is illustrated in Figure 5.1. It also shows the feasible solution for each trial.

In the genetic algorithm, there are two parameters, which are listed in the following:

- number of allocation strings in a set,
- termination constant.

The relationship between the strings in the final set and the number of allocation strings in a set is already discussed. The number of strings in a set is also highly related to the number of workstations in the problem and the execution time of the search algorithm. The number of strings in a set is directly proportional to the number of workstation in the problem. For an increase in the problem size, the number of strings in a set should be increased as well, to prevent duplication strings as in the problem mentioned before. The result of those tests, generated for the same numerical example, are shown in the following table and illustrated in Figure 5.2 and 5.3. All of the tests obtained the same
Feasible Solution Obtained at Trial 5

Average Cost and Expected Cost

Trial Number

Average Cost of Production — Expected Cost for Feasible Solution

Figure 5.1. Trial Number vs. Average Cost of Production.
optimal allocation of workstations showed in Table 5.4.

Table 5.8. Result of Tests with Different Number of Strings in a Set.

<table>
<thead>
<tr>
<th>Number of Strings in a Set</th>
<th>Number of Duplication Strings in Final Trial</th>
<th>Percentage of Duplication Strings in Final Trial</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>100%</td>
<td>86 Seconds</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>78%</td>
<td>110 Seconds</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>27%</td>
<td>132 Seconds</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>15%</td>
<td>168 Seconds</td>
</tr>
</tbody>
</table>

In Figure 5.2, the graph shows that the number of strings in a set is indirectly proportional to the percentage of duplication of strings in the final trial. For the case of 7 strings in a set, the percentage of duplication is 100% (as previous discussed). In Figure 5.3, the number of strings in a set is directly proportional to the execution time of the search algorithm. The search time is lower for a set with less strings, but it is limited by string duplication in the final set. Furthermore, a search with a small number of strings may generate a feasible solution which is far from the optimal. The optimal number of strings for this problem is 11, which has a low 27% percentage of duplication and an execution time of 132 seconds in an IBM comparable 486 computer with execution speed of 66 MHz.

The second control parameter is the termination constant (α). It generates the termination factor (TF) by Equation (34), which controls the termination of search
Figure 5.2. Number of Strings in a Set vs. Percentage of Duplication of Strings.
Figure 5.3. Number of Strings in a Set vs. Execution Time.
algorithm. A few tests are performed with different values of termination constant. The results are shown in Table 5.9 and in Figures 5.4 and 5.5.

Table 5.9. Result of Tests with Different Values of Termination Constant.

<table>
<thead>
<tr>
<th>Termination Constant</th>
<th>Termination Factor</th>
<th>Optimal Achieve Probability</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>3</td>
<td>0.40</td>
<td>43 Seconds</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0.80</td>
<td>60 Seconds</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1.00</td>
<td>93 Seconds</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>1.00</td>
<td>132 Seconds</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>1.00</td>
<td>167 Seconds</td>
</tr>
</tbody>
</table>

As shown, the value of the termination constant is directly proportional to the optimal achieve probability. In the case where the termination constant equals 0.5, the optimal achieve probability is only 0.40. When the termination constant equals 1, the optimal achieve probability is 0.80. For those cases with a termination constant equal to and larger than 2, the optimal achieve probability is 1.00. Figure 5.5 shows the value of the termination constant as directly proportional to the execution time; the optimal value for this 5 workstations allocation problem is 2.

With the two control parameters, 11 strings in a set and a termination constant of 2, a few tests are performed to obtain the average execution time. The results are shown in Table 5.10.
Figure 5.4. Termination Constant vs. Optimal Achieve Probability.
Figure 5.5. Termination Constant vs. Execution Time.
Table 5.10. Tests Result of Execution Time.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Optimal Obtained in Trial Number</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>93 seconds</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>86 seconds</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>168 seconds</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>72 seconds</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>88 seconds</td>
</tr>
</tbody>
</table>

In Table 5.10, five tests are performed to obtain the average execution time of the genetic algorithm in an IBM comparable 486 computer with execution speed of 66 MHz. The average execution time is 101.4 seconds.

5.5 Other Numerical Examples

The genetic algorithm is applied to three more example with 10, 15, and 20 workstations. For the second numerical examples with 10 workstations (Appendix D), Table 5.11 shows the results for different numbers of strings in a set. The test with 17 strings has a 12% duplication of strings in the final set, which is acceptable for a search problem of this size. Table 5.12 shows the results of testing different values of termination constant for this 10 workstation allocation problem.

In Table 5.12, the feasible solution obtained from the tests with the termination constant values of 2, 3, 4, and 5 are the same. When the value increases to 6, a new improved feasible solution is obtained. This solution stays the same when the value is increased
to 7. Therefore, the termination factor is 70. For this test, the feasible solution is obtained at trial 74, and the search is terminated at trial 143. The execution time of the search is 20 minutes and 57 seconds.

Table 5.11. Result of Tests with Different Number of Strings in a Set.

<table>
<thead>
<tr>
<th>Number of Strings in a Set</th>
<th>Number of Duplication Strings in Final Trial</th>
<th>Percentage of Duplication Strings in Final Trial</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>8</td>
<td>73%</td>
<td>315 Seconds</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>62%</td>
<td>351 Seconds</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>33%</td>
<td>402 Seconds</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>12%</td>
<td>481 Seconds</td>
</tr>
</tbody>
</table>

Table 5.12. Result of Tests with Different Values of Termination Constant.

<table>
<thead>
<tr>
<th>Termination Constant</th>
<th>Termination Factor</th>
<th>Feasible Solution</th>
<th>Expected Cost</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>20</td>
<td>9 7 4 10 1 3 8 5 6 2</td>
<td>56.58</td>
<td>8 Min. 1 Sec.</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>9 7 4 10 1 3 8 5 6 2</td>
<td>56.58</td>
<td>9 Min. 23 Sec.</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>9 7 4 10 1 3 8 5 6 2</td>
<td>56.58</td>
<td>11 Min. 30 Sec.</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>9 7 4 10 1 3 8 5 6 2</td>
<td>56.58</td>
<td>13 Min. 52 Sec.</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>7 9 5 10 1 4 2 3 8 6</td>
<td>54.91</td>
<td>16 Min. 48 Sec.</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>7 9 5 10 1 4 2 3 8 6</td>
<td>54.91</td>
<td>20 Min. 57 Sec.</td>
</tr>
</tbody>
</table>
A few test is generated with the about parameters to determine the average execution time. The results of the tests are listed in Table 5.13. Four out of five tests obtain the same feasible solution. The average execution time for the four successful tests is 22 minutes and 29 seconds. The feasible solution of workstations allocation and the number of kanbans for this allocation is shown in Table 5.14.

Table 5.13. Tests Result of Execution Time.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Trial Obtained the Feasible Solution</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>74</td>
<td>20 Min. 57 Sec.</td>
</tr>
<tr>
<td>2</td>
<td>92</td>
<td>26 Min. 11 Sec.</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>12 Min. 43 Sec.</td>
</tr>
<tr>
<td>4</td>
<td>106</td>
<td>30 Min. 04 Sec.</td>
</tr>
<tr>
<td>5</td>
<td>68 (Non-success test)</td>
<td>19 Min. 16 Sec.</td>
</tr>
</tbody>
</table>

Table 5.14. Allocation of Workstations and Number of Kanbans for the Example.

<table>
<thead>
<tr>
<th>Work Stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>No. of Kanban</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

For this example of 10 workstations, the feasible solution is a near-optimal (perhaps the optimal) solution. It is impossible to guarantee that the feasible solution is an optimal solution. However, it is possible to compare it with the result generated by a second search algorithm, simulated annealing.
For the example of allocating 15 workstations (Appendix E), the control parameters used are:

- the number of strings in a set is 21,
- the termination constant is 8, which generates a termination factor of 120.

The average execution time of the search is 46 minutes and 32 seconds.

For the example of allocating 20 workstations (Appendix F), the control parameters used are:

- the number of strings in a set is 23,
- the termination constant is also 8, which generates a larger termination factor of 160.

The average execution time of the search is 1 hour, 32 minutes and 4 seconds.
CHAPTER 6
SIMULATED ANNEALING

6.1 Mechanics of Simulated Annealing

Before applying simulated annealing to the numerical example developed in Section 4.1, a brief discussion of the mechanics is presented. Simulated annealing has three components:

- acceptance criteria,
- generation of neighbourhood string,
- cooling schedule.

Simulated annealing is a probabilistic hill jumping combinatorial optimization technique. The hill jumping feature allows increased objective function (for a minimization problem) in a controlled manner, to prevent trapping in a local minimum. Simulated annealing not only searches for improvement in objective function, it also accepts a solution which has no improvement control over a probability. This acceptance probability is controlled by an acceptance criterion. There are many different acceptance criteria in simulated annealing for different kinds of problems. One of the most common and simple ones is the Metropolis criterion, an algorithm developed by Metropolis et al. (1953) for simulating the evolution of a solid in a heat bath to thermal equilibrium.

The Metropolis criterion is shown in the following equation.
\[ P \{ \text{accept } i + 1 \} = \begin{cases} 
1 & \text{for } f(i) \geq f(i + 1) \\
\exp \left( \frac{f(i) - f(i + 1)}{c} \right) & \text{for } f(i) < f(i + 1) 
\end{cases} \] (35)

In Equation (35), \( P \{ \text{accept } i + 1 \} \) is the probability of accepting the solution \( i + 1 \) with \( f(i) \) and \( f(i + 1) \) are the objective functions of solutions \( i \) and \( i + 1 \). The probability of accepting an improved solution is 1. A solution without any improvement is accepted in a probability. Control parameter \( c \) which will be defined later in this chapter.

The new string generated by simulated annealing must be a neighbour of the previous string. For the proposed workstation allocation production, the best method to generate a new neighbour is mutation (also known as permutation), as introduced in the discussion of genetic algorithm.

The search of simulated annealing is controlled by a set of parameters. These parameters are called the cooling schedule. They are:

- an initial value of the control parameter \( c_0 \),
- a decrement function for decreasing the value of the control parameter \( c \),
- a termination criterion \( \alpha \) control the termination of the search algorithm,
- a finite number of trial (length) \( L \) at each value of the control parameter \( c \).

The initial value of the control parameter \( c_0 \) controls the acceptance probability of strings.
without improvement. It should be large enough to allow jumps over the highest peaks in the search space. The acceptance probability for this initial control parameter $c_0$ should be over 80%. Its value will be defined later in this chapter.

The decrement function is controlled by the following equation, with a decrement constant $dc$.

$$c_{i+1} = dc \times c_i$$  \hspace{1cm} (36)

The decrement of the control parameter reduces the probability of accepting a string which has no improvement in the objective function. In most of the previous research, the value of $dc$ is a real number in between 0.8 and 0.99.

The termination of simulated annealing is the same as in the genetic algorithm. A termination constant is used to control the termination factor, which could be determined by Equation (34).

For each value of the control parameter $c_i$, a finite number of trial (length) $L_i$ should be performed. This trial limit $L_i$ is highly related to the value of the control parameter $c_i$. It is calculated by the following equation, where $L_0$ is the initial value of $L$.

$$L_i = \text{INT} \left( L_0 + L_0 \times (c_0 - c_i) \times 0.5 \right)$$ \hspace{1cm} (37)

The trials limit increases with the search stage $i$. It is because the control parameter keep
decreasing with the search stage. The probability of accepting string without improvement (probability of hill jumping) is also decreases with the search stage. To increase this probability, the number of trials for the later search stages have to be increased.

6.2 Model Application

The application of simulated annealing to the workstation allocation problem could be divided into the following steps:

1. Input the number of workstations $N$, the processing rate for each workstation in different positions, the failure rate and the recovery rate for each workstation, and other required data.

2. Randomly generate a set of legal allocation string.

3. Evaluate the performance of each string with the mathematical model in Chapter 3.

4. Define the best string in the set.

5. Generate a new set of strings by simulated annealing (mutation).

6. Check that each new string is legal, replace an illegal string by generating a new legal string.

7. Re-evaluate each new string using the mathematical model.

8. Accept or reject the new string using the acceptance criterion.

9. Repeat Steps 4, 5, 6, 7, and 8 until the number of trials reaches $L_t$.

10. Update (decrease) the value of control parameter $c_t$. 

90
11. Repeat Steps 4 through 10 until no more improvement is achieved.

The first four steps are the same as in genetic algorithm. In the search, a set of strings is used instead of a single string. This can improve the ability of search. In Step 5, a new set of neighbourhood strings is generated by mutation (as discussed in the previous chapter). Step 6 and 7 are also the same for genetic algorithm. Step 8 is presented in the beginning of this chapter.

Steps 4, 5, 6, 7, and 8 are repeated until the number of trials reaches the trial limit \( L_t \). After that, the control parameter is decreased. Step 4 through step 10 are also repeated, until no string with a better performance function is available. The termination of the search algorithm is controlled by a termination factor as in genetic algorithm.

6.3 Application of Simulated Annealing to the Numerical Example

Simulated annealing is applied to the example developed in Chapter 4. The required data is given in Section 4.1. A computer program in FORTRAN language, SA.FOR (see Appendix G), is developed to solve the problem of workstations allocation by simulated annealing. It shares the same data file, W5.DAT (Appendix B), of 5 workstations allocation problem with genetic algorithm.

6.4 Discussion of Result for the Numerical Example of Allocating 5 Workstations

The search of simulated annealing is controlled by the cooling schedule. It has four
parameters:

- an initial value of the control parameter $c_0$,
- a decrement constant $dc$ for decreasing the value of the control parameter $c_i$ showed in Equation (36),
- a termination criterion $\alpha$ control the termination of the search algorithm showed in Equation (34),
- an initial value of trial limit $L_0$ showed in Equation (37).

And there is one more parameter. This is the number of strings in a set. This number, for simple simulated annealing, is 1. By increasing the number of strings in a set, the search space will be increased. A set of strings is searched simultaneously and independently, except for the comparison of objective functions to define the best one. A few tests are performed with a different number of strings in the set. All the parameters of the cooling schedule are the same for these tests, with values $c_0=1.5$, $dc=0.9$, $\alpha=1$, and $L_0=5$. The result is shown in the following table and figures.

<table>
<thead>
<tr>
<th>Number of Strings in Set</th>
<th>Optimal Achieve Probability</th>
<th>Total Number of Trial</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2</td>
<td>48</td>
<td>26 Seconds</td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>136</td>
<td>75 Seconds</td>
</tr>
<tr>
<td>3</td>
<td>0.9</td>
<td>144</td>
<td>85 Seconds</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>192</td>
<td>115 Seconds</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>240</td>
<td>139 Seconds</td>
</tr>
</tbody>
</table>
From Figure 6.1, a set with more than 4 strings has the optimal achieve probability of 1.0 for the given parameter values. A set of less than 4 strings has the optimal achieve probability lower than 1.0. Thus, the number of strings in the set used should be larger than 4. However, the optimal achieve probability of those sets lower than 1.0 could be improved by adjusting the parameters. For the number of strings in a set equal to 1, the optimal achieve probability for the above test is 0.2. By increasing the termination constant $\alpha$ to 2, and the initial trial limit $L_0$ to 10, the optimal achieve probability will be increased to 1.0. The execution time is 207 seconds, which is longer than for a set with a number of strings equal 5. This is because search is conducted over a population of strings (5) simultaneously and independently, reducing the probability of locating a false peak with less trials. The suggested number of strings in a set is 5 for this numerical example.

Figure 6.2 shows the number of trials for each test with the given cooling schedule. The number of strings in the set is directly proportional to the number of trials. Figure 6.3 illustrates that the execution time is also directly proportional to the number of strings in the set.

For the first parameter in the cooling schedule, the initial value of the control parameter $c_0$. Some tests are generated with different values of the initial control parameter $c_0$. The results are shown in the following table and figures.
Figure 6.1. Number of Strings in a Set vs. Optimal Achieve Probability.
Figure 6.2. Number of Strings in a Set vs. Total Number of Trials.
Figure 6.3. Number of Strings in a Set vs. Execution Time.
Table 6.2. Result of Tests with Different Value of Initial Control Parameter $c_0$.

<table>
<thead>
<tr>
<th>Value of Initial Control Parameter</th>
<th>Probability of Improved String Accepted</th>
<th>Probability of Non-improved String Accepted</th>
<th>Number of Trials</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.31707</td>
<td>0.39286</td>
<td>205</td>
<td>108 Seconds</td>
</tr>
<tr>
<td>1.25</td>
<td>0.31364</td>
<td>0.45695</td>
<td>220</td>
<td>114 Seconds</td>
</tr>
<tr>
<td>1.5</td>
<td>0.31667</td>
<td>0.45732</td>
<td>240</td>
<td>138 Seconds</td>
</tr>
<tr>
<td>2.0</td>
<td>0.33333</td>
<td>0.49412</td>
<td>255</td>
<td>137 Seconds</td>
</tr>
<tr>
<td>2.5</td>
<td>0.36981</td>
<td>0.59880</td>
<td>265</td>
<td>157 Seconds</td>
</tr>
</tbody>
</table>

In Figure 6.4, the probabilities of both improved and non-improved string accepted are directly proportional to the value of initial control parameter. The probability of improved string accepted is always lower than the probability of non-improved string accepted. This means the probability of generating an improved string is much lower than the probability of generating a non-improved string. Figure 6.5 shows the number of trials as directly proportional to the value of the initial control parameter; this parameter is directly related to the trial limit $L_i$ in Equation (37), which in turn controls the search algorithm. Figure 6.6 shows the value of initial control parameter as directly proportional to the execution time.

The ideal value of the initial control parameter is the one which has the highest probability of non-improved string accepted, since this probability is directly proportional to the value of initial control parameter. A larger value is preferred, but it will increase
Figure 6.4. Value of Initial Control Parameter vs. Probability of Strings Accepted.
Figure 6.5. Value of Initial Control Parameter vs. Total Number of Trials.
Figure 6.6. Value of Initial Control Parameter vs. Execution Time.
the execution time of the search. Therefore, the value of the initial control parameter used for this example is 1.5.

For varied values of the termination constant $\alpha$, a few tests are conducted. The number of strings in the set is 5. The cooling schedule is the same as above except for the termination constant. Table 6.3 and Figures 6.7 and 6.8 present the results.

Table 6.3. Result of Tests with Different Values of Termination Constant.

<table>
<thead>
<tr>
<th>Termination Constant</th>
<th>Termination Factor</th>
<th>Optimal Achieve Probability</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>2</td>
<td>0.40</td>
<td>60 Seconds</td>
</tr>
<tr>
<td>0.6</td>
<td>3</td>
<td>0.80</td>
<td>86 Seconds</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1.00</td>
<td>138 Seconds</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1.00</td>
<td>271 Seconds</td>
</tr>
</tbody>
</table>

In Figure 6.7, the value of the termination constant is directly proportional to the optimal achieve probability. When the termination constant equals 0.4, the optimal achieve probability is only 0.40. When the termination constant is 0.6, the optimal achieve probability is 0.80. For those cases where the termination constant is equal to and greater than 1, the optimal achieve probability is 1.00. Figure 6.8 demonstrates that the termination constant value is also directly proportional to the execution time. The optimal value of the termination constant for this 5 workstation allocation problem is 1.
Figure 6.7. Termination Constant vs. Optimal Achieve Probability.
Figure 6.8. Termination Constant vs. Execution Time.
The value of the decrement constant \( dc \) is defined by a few tests. The results are illustrated in the following table and figures.

<table>
<thead>
<tr>
<th>Value of Decrement Constant</th>
<th>Probability of Improved String Accepted</th>
<th>Probability of Non-improved String Accepted</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.99</td>
<td>0.41143</td>
<td>0.66019</td>
<td>105 Seconds</td>
</tr>
<tr>
<td>0.90</td>
<td>0.33878</td>
<td>0.49383</td>
<td>150 Seconds</td>
</tr>
<tr>
<td>0.80</td>
<td>0.35000</td>
<td>0.47436</td>
<td>153 Seconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value of Decrement Constant</th>
<th>Trials Limit ( L_t ) for Search Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.99</td>
<td>5</td>
</tr>
<tr>
<td>0.90</td>
<td>5</td>
</tr>
<tr>
<td>0.80</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 6.9 shows that the probabilities of both improved string non-improved string accepted are directly proportional to the decrement constant. Figure 6.10 shows that the execution time is indirectly proportional to the decrement constant. In Figure 6.11, the trials limit for each search stage are plotted against the search stage. The decrement constant equal to 0.99 is too close to 1, giving no change in the trials limit. Their relationships are found in Equation (36) and (37). For the decrement constants equal to

104
**Figure 6.9.** Value of Decrement Constant vs. Probability of Strings Accepted.
Figure 6.10. Value of Decrement Constant vs. Execution Time.
Figure 6.11. Trial Number vs. Trial Limit for Different Value of Decrement Constant.
0.9 and 0.8, the trials limit is directly proportional to the search stage. The decrement constants equal to 0.99 has the shortest execution time, and highest probability of non-improved string accepted. This could improve the search, but the trials limit of each search stage remains unchanged. The probability of non-improved string accepted for the later stage will be decreased. Thus, this value of decrement constant is not preferred for this example. Instead, a decrement constant equal to 0.9 is used, since it has a short execution time, high probability of non-improved string accepted, and the trials limit changes with the search stage.

The last parameter in the cooling schedule is the initial value of trial limit \( L_0 \). A few tests are generated with different values of initial trial limits. The results are given in Tables 6.6 and 6.7, and Figures 6.12, 6.13, 6.14, and 6.15.

<table>
<thead>
<tr>
<th>Trials Limit</th>
<th>Optimal Achieve Probability</th>
<th>Total Number of Trial</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.6</td>
<td>115</td>
<td>72 Seconds</td>
</tr>
<tr>
<td>3</td>
<td>0.8</td>
<td>170</td>
<td>103 Seconds</td>
</tr>
<tr>
<td>4</td>
<td>0.9</td>
<td>205</td>
<td>127 Seconds</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>245</td>
<td>150 Seconds</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>280</td>
<td>166 Seconds</td>
</tr>
</tbody>
</table>

Table 6.6. Result of Tests with Different Value of Initial Trials Limit.
Table 6.7. Trials Limit $L_i$ for Different Value of Initial Trials Limit.

<table>
<thead>
<tr>
<th>Initial Trials Limit</th>
<th>Trials Limit $L_i$ for Search Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 2 3 3 3 3 3 4</td>
</tr>
<tr>
<td>3</td>
<td>3 3 4 4 5 5 5 5</td>
</tr>
<tr>
<td>4</td>
<td>4 5 5 6 6 7 7 7</td>
</tr>
<tr>
<td>5</td>
<td>5 6 6 7 8 8 8 9</td>
</tr>
<tr>
<td>7</td>
<td>7 8 9 10 11 11 -- --</td>
</tr>
</tbody>
</table>

Figure 6.12 shows the optimal achieve probability as directly proportional to the initial trials limit. The limits equal to or larger than 5 have the optimal achieve probability of 1.0. Limits less than 5, have an optimal achieve probability lower than 1.0, simply because the number of trials is reduced with the decrease of trial limit, and this limit is controlled by the initial trial limit. Figure 6.13 shows the number of trials as directly proportional to the initial trials limit. The execution time is also directly proportional to the initial trials limit. This is illustrated in Figure 6.14.

In Figure 6.15, the trials limit for each search stage is plotted against the search stage for the five initial trials limits. The graph shows that the trials limit is increased with the search stage, and the highest initial trial limit has the highest set of trial limits. Their relationship is found in Equation (37). The trials limit used for this numerical example is 5. It has the shortest execution time and its optimal achieve probability is 1.0.
Figure 6.12. Initial Trial Limit vs. Optimal Achieve Probability.
Figure 6.13. Initial Trial Limit vs. Total Number of Trials.
Figure 6.14. Initial Trial Limit vs. Execution Time.
Figure 6.15. Trial Number vs. Trial Limit for Different Value of Initial Trial Limit.
With the ideal number of strings in a set and all other parameters of the cooling schedule defined, a number of tests are performed to obtain the average execution time of the simulated annealing for the numerical example. Tests are performed by an IBM comparable 486 computer with an execution speed of 66 MHz. The results are shown in the following table. The average execution time is 148.2 seconds, and all five tests obtained the same optimal solution defined by genetic algorithm.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Stage &amp; Trial Obtained the Optimal</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Trial 7, Stage 3</td>
<td>152 Seconds</td>
</tr>
<tr>
<td>2</td>
<td>Trial 2, Stage 2</td>
<td>138 Seconds</td>
</tr>
<tr>
<td>3</td>
<td>Trial 3, Stage 3</td>
<td>150 Seconds</td>
</tr>
<tr>
<td>4</td>
<td>Trial 3, Stage 4</td>
<td>163 Seconds</td>
</tr>
<tr>
<td>5</td>
<td>Trial 4, Stage 2</td>
<td>138 Seconds</td>
</tr>
</tbody>
</table>

### 6.5 Other Numerical Examples

The simulated annealing is applied to three more numerical examples with 10, 15, and 20 workstations (Appendix D, E, F). These are the same examples used by the genetic algorithm in Chapter 5. For the example of allocating 10 workstation, the number of strings used is 7, and parameters for the cooling schedule are:

- initial value of the control parameter $c_0$ is 4
- decrement constant $dc$ is 0.95
• termination criterion $\alpha$ is 1.5
• initial value of trial limit $L_0$ is 10

These parameters are determined by several tests of the search algorithm. Table 6.9 shows the execution time of the tests. The average search execution time is 27 minutes and 28 seconds. As shown in Table 6.10, the feasible solution obtained is the same as the one gathered by genetic algorithm. The number of kanbans for this allocation is also shown in Table 6.10.

Table 6.9. Tests Result of Execution Time.

<table>
<thead>
<tr>
<th>Test Number</th>
<th>Stage &amp; Trial Obtained the Optimal</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Trial 8, Stage 12</td>
<td>28 Min. 14 Sec.</td>
</tr>
<tr>
<td>2</td>
<td>Trial 5, Stage 7</td>
<td>22 Min. 32 Sec.</td>
</tr>
<tr>
<td>3</td>
<td>Trial 16, Stage 13</td>
<td>30 Min. 29 Sec.</td>
</tr>
<tr>
<td>4</td>
<td>Trial 3, Stage 9</td>
<td>24 Min. 27 Sec.</td>
</tr>
<tr>
<td>5</td>
<td>Trial 11, Stage 15</td>
<td>31 Min. 36 Sec.</td>
</tr>
</tbody>
</table>

Table 6.10. Allocation of Workstations and Number of Kanbans for the Example.

<table>
<thead>
<tr>
<th>Work Stage</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>No. of Kanban</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

For the example of allocating 15 workstations, the number of strings in a set is 9
and parameters for the cooling schedule are:

- initial value of the control parameter $c_0$ is 4
- decrement constant $dc$ is 0.95
- termination criterion $\alpha$ is 2
- initial value of trial limit $L_0$ is 15

The average execution time of the search is 1 hour 9 minutes and 46 seconds.

For the example of allocating 20 workstations, the number of strings in a set is 11 and parameters for the cooling schedule are:

- initial value of the control parameter $c_0$ is 4
- decrement constant $dc$ is 0.95
- termination criterion $\alpha$ is 2
- initial value of trial limit $L_0$ is 20

The average execution time of the search is 1 hour 52 minutes and 10 seconds.

In the next chapter, the results of both search algorithms are compared, and the conclusion of the research is presented.
CHAPTER 7

CONCLUSION

7.1 Comparison of the Two Search Algorithms

Genetic algorithm based primarily on random number generation, searches through the solution space by a set of strings instead of a single string. New strings are generated from the superior portion of previous sets by reproduction, crossover, and a low probability of mutation. The two control parameters are the number of strings in a set and the termination factor.

Simulated annealing is based on probabilistic hill jumping technique. It searches through a solution space by the hill jumping technique with an acceptance criterion. A new string is generated by the neighbourhood approach through mutation. This research proposes that sets of strings are searched simultaneously and independently. The objective functions of strings are compared to define the superior one. This search algorithm is controlled by a cooling schedule, whose parameters are required to determine to fit the search problem.

In previously presented testing of a 5 workstation allocation example, these two search algorithms both obtained the optimal allocation by the deterministic method. For the problem with 10 workstations, both search algorithms did acquire the same feasible solution. However, the genetic algorithm successfully gained the solution in four out of
five tests. The simulated annealing reached the solution in all five tests. For the example with 15 workstations, the two search algorithms obtained the same feasible solution. For the problem of allocating 20 workstations, the genetic algorithm successfully gained an identical solution in four out of five tests. The simulated annealing found the same solution in all five tests. The choice of control parameters is very important in order to obtain the optimal, or near optimal solution. As shown in Table 7.1 and Figure 7.1, the execution time of the genetic algorithm is shorter than that of the simulated annealing for all examples. However, it should be noticed that the performance of these two search algorithms is case dependent.

Table 7.1. Average Execution Time for the Four Examples.

<table>
<thead>
<tr>
<th>Number of Workstations</th>
<th>Genetic Algorithm</th>
<th>Simulated Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1 min. 41 sec.</td>
<td>2 min. 28 sec.</td>
</tr>
<tr>
<td>10</td>
<td>20 min. 57 sec.</td>
<td>27 min. 28 sec.</td>
</tr>
<tr>
<td>15</td>
<td>46 min. 32 sec.</td>
<td>1 hr. 9 min. 46 sec.</td>
</tr>
<tr>
<td>20</td>
<td>1 hr. 22 min. 19 sec.</td>
<td>1 hr. 52 min. 10 sec.</td>
</tr>
</tbody>
</table>

Genetic algorithm has two control parameters:

- number of strings in a set,
- termination factor.

Both control parameter are highly related to the number of workstations in the search problem. They can be easily determined, which greatly increases the capability for the application of genetic algorithm.
Figure 7.1. Number of Workstations vs. Execution Time.
Simulated annealing's cooling schedule has five parameters including the number of strings in a set. They are all related to the number of workstations in the problem and the value of the objective function. The five parameters are very complicated and are related to each other. For the best result with this search algorithm, each parameter must be finely chosen. As well, there are many distinct acceptance criteria for different kinds of problems. The appropriate acceptance criterion should be used. All these factors make the simulated annealing more difficult to apply. A summary for the comparison of genetic algorithm and simulated annealing is presented in Table 7.2.

Table 7.2. Comparison of Genetic Algorithm and Simulated Annealing.

<table>
<thead>
<tr>
<th></th>
<th>Genetic Algorithm</th>
<th>Simulation Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution Time</strong></td>
<td>Shorter</td>
<td>Longer</td>
</tr>
<tr>
<td><strong>Number of Control Parameter</strong></td>
<td>2 (Easy to Define)</td>
<td>3 (Complicated and Hard to Define)</td>
</tr>
<tr>
<td><strong>Control of Search</strong></td>
<td>Less Control</td>
<td>Better Control with Appropriate Control Parameter</td>
</tr>
<tr>
<td><strong>Ability to Obtain Optimal</strong></td>
<td>Good Guarantee of Success</td>
<td>Better Guarantee of Success</td>
</tr>
<tr>
<td><strong>Capability of Application</strong></td>
<td>Easier to Apply</td>
<td>Harder to Apply</td>
</tr>
</tbody>
</table>
7.2 Conclusion

The proposed workstation allocation problem is a combinatorial optimization problem. The objective is to determine the optimal allocation of a finite number of unreliable workstations to a finite number of work stages in a pull production line. To determine the allocation, the optimal number of kanbans for each work stage must first be defined.

The proposed problem is formulated by a mathematical model in Chapter 3. Two search algorithms, genetic algorithm and simulated annealing, are applied to the problem in Chapters 5 and 6. The results of both search algorithms are studied in Chapters 5 and 6, and compared in Section 7.1.

In the initial set of strings, two allocations of workstations are configurated according to the bowl phenomenon. However, they are not the optimal (or near-optimal) solutions. The results of the two search algorithms indicate that for a pull production line with multiple unreliable workstations, the optimal (or near-optimal) arrangement of the workstations does not agree with the bowl phenomenon. This is due to the JIT operational characteristic of the proposed system.

Genetic algorithm is simpler and more easily applied, with two easily determined control parameters. Its search execution time for the proposed problem is shorter. Simulated annealing is complicated and its application more difficult, having four parameters in the cooling schedule which is much harder to determine. However, with an appropriate cooling schedule, the simulated annealing performs much better than the genetic
algorithm. Simulated annealing has a better guarantee of success, since it has more control in the search.

Both genetic algorithm and simulated annealing are stochastic optimization methods. They are easier to apply than the deterministic approach. The time required for the application and search is also much less than for the deterministic method, especially for large problems. The two search algorithms do solve the workstation allocation problem with the mathematical model. The success of the search is highly related to the termination factor, which determines when the search algorithm will stop. The solution generated is not guaranteed to be the optimal solution. It is, rather, a near-optimal allocation of workstations.

7.3 Future Work

This research could be expanded in several areas.

1. The mathematical model could be developed to consider a production line for multiple products.

2. Instead of a single workstation in each work stage, multiple workstations are possible.

3. The proposed problem has the same number of workstations and work stages. The case for excessive workstations could be studied.

4. The relationship of the cooling schedule for simulated annealing with the problem could be determined.
5. In simulated annealing, there are other acceptance criteria which could be applied to the problem.

6. Other stochastic search approaches could be applied to the proposed problem, such as random search.

7. Comparing the two stochastic search algorithms with the traditional deterministic method may be performed.
REFERENCES


23. Magazine, M. J., and Silver, G. L., "Heuristics for Determining Output and


Appendix A. Listing of GA.FOR.
PROGRAM OF GENETIC ALGORITHM

INTEGER i,j,k,l,m,n,x,pw(15),ks(15),ns,nmu
INTEGER s(20,15),relo(15),itemp,cent,tri,maxs,mins
INTEGER cx1,cx2,cs,ce,cn,mu1,mu2,mut,termco
REAL c1,c3,d,f(15),r(15),u(15,15),hcs(15),scs(15)
REAL rel(15),temp,re(15),vtemp(15),URAND
REAL tcx,ct1,ct2,ct3,tc(20),avertc,maxtc,mintc,last
x=13850
termco=8
ns=17
probmu=1000

DATA INPUT

OPEN(15,FILE='W5.DAT')
READ(15,*)n,c1,c3,d,(f(i),i=1,n),(r(i),i=1,n)
DO 400 i=1,n
   READ(15,*)(u(i,j),j=1,n)
400 CONTINUE
READ(15,*)(hcs(i),i=2,n),(scs(i),i=2,n)
CLOSE(15)

GENERATE ALLOCATION STRINGS

DO 410 i=1,n
   rel(i)=r(i)/(f(i)+r(i))
   relo(i)=i
   re(i)=rel(i)
410 CONTINUE

DO 420 k=1,n-1
   DO 430 l=1,n-k
      IF (rel(l+1).LT.re(i)) THEN
         temp=rel(l)
         rel(l)=rel(l+1)
         rel(l+1)=temp
         itemp=relo(l)
         relo(l)=relo(l+1)
         relo(l+1)=itemp
      END IF
430 CONTINUE
420 CONTINUE

DO 440 l=1,n
   s(1,l)=relo(l)
   s(2,n+1-l)=relo(l)

129
440 CONTINUE  
cent=INT(n/2)+1  
DO 445 m=3,4  
s(m,cent)=relo(1)  
DO 450 k=1,cent-1  
s(m,cent-k)=relo(k*2)  
450 CONTINUE  
DO 460 k=1,n-cent  
s(m,cent+k)=relo(k*2+1)  
460 CONTINUE  
DO 470 i=1,n  
vtemp(i)=relo(i)  
470 CONTINUE  
DO 480 i=1,n  
relo(i)=vtemp(n+1-i)  
480 CONTINUE  
445 CONTINUE  
DO 490 l=5,ns  
DO 500 m=1,n  
520 temp=INT(n*URAND(x))+1  
DO 510 i=1,m-1  
x=x+1  
IF (temp.EQ.s(l,i)) GO TO 520  
510 CONTINUE  
s(l,m)=temp  
500 CONTINUE  
490 CONTINUE  

C EVALUATION OF EACH ALLOCATION STRING  
C  
tri=0  
last=999999  
nmu=0  
800 tri=tri+1  
DO 530 l=1,ns  
DO 540 m=1,n  
pw(m)=s(l,m)  
540 CONTINUE  
CALL COST(n,c1,c3,d,pw,f,r,u,hcs,scs,tcx,ct1,ct2,ct3,ks)  
tc(l)=tcx  

C PRINT*, ’Total Cost For String’,l,’ is’,tc(l)  
C PRINT*,l,’ is’,s(l,m),m=1,n  
C
CONTINUE
  avertc = sum(ns, tc)/ns
PRINT 87, tri, avertc
87 FORMAT(’Average cost of trial ’, i3, ’ is’, F13.7)
C
C DEFINE THE BEST ALLOCATION STRING
C
max tc = tc(1)
max s = 1
mintc = tc(1)
mins = 1
DO 550 l = 2, ns
  IF (tc(l) .GT. max tc) THEN
    max tc = tc(l)
    max s = 1
  ELSE IF (tc(l) .LT. mintc) THEN
    mintc = tc(l)
    mins = 1
  END IF
550 CONTINUE
IF (mintc .EQ. last) THEN
  term = term + 1
  IF (term .EQ. (termco * n)) GO TO 900
ELSE
  term = 1
  last = mintc
PRINT 83, (s(mins, m), m = 1, n), tc(mins)
83 FORMAT(’The Best Allocation Is’, i3, ’ With TC Of’, F13.7)
END IF
C
C GENETIC ALGORITHM
C
C REPRODUCTION
C
DO 560 m = 1, n
  s(maxs, m) = s(mins, m)
  temp = s(ns, m)
  s(ns, m) = s(maxs, m)
  s(maxs, m) = temp
560 CONTINUE
C
C PARTIALLY MAPPED Crossover
C
  cn = INT(ns/2)
DO 570 k=1,cn
   x=x+1
   cx1=INT((ns-1)*URAND(x))+1
   x=x+1
   cx2=INT((ns-1)*URAND(x))+1
IF (cx1.EQ.cx2) go to 580
DO 590 m=1,n
   itemp=s(cx1,m)
   s(cx1,m)=s(cx2,m)
   s(cx2,m)=itemp
590    CONTINUE
570    CONTINUE
DO 595 k=1,ns-2,2
   x=x+1
   cs=INT(n*URAND(x))+1
   x=x+1
   ce=INT(n*URAND(x))+1
IF (cs.GT.ce) THEN
   itemp=cs
   cs=ce
   ce=itemp
END IF
DO 600 m=cs,ce
   itemp=s(l,m)
   DO 610 k=1,n
      IF (s(l,k).EQ.s(l+1,m)) THEN
         s(l,k)=s(l,m)
      END IF
      IF (s(l+1,k).EQ.s(l,m)) THEN
         s(l+1,k)=s(l+1,m)
      END IF
610    CONTINUE
s(l,m)=s(l+1,m)
s(l+1,m)=itemp
600    CONTINUE
595    CONTINUE
C
C    MUTATION
C
x=x+1
mut=INT(probm*URAND(x))
IF (mut.GT.1) GO TO 800
nmu=nmu+1
x=x+1
mu1 = INT(n*URAND(x)) + 1
620  x = x + 1
    mu2 = INT(n*URAND(x)) + 1
    IF (mu1.EQ.mu2) GO TO 620
    x = x + 1
    mu = INT((ns-1)*URAND(x)) + 1
    itemp = s(mu,mu1)
    s(mu,mu1) = s(mu,mu2)
    s(mu,mu2) = itemp
    GO TO 800

C
    FEASIBLE SOLUTION OBTAINED AND OUTPUT
C
900    PRINT*, ’The Feasible Solution Obtained In Trial Number’,tri
    PRINT*, ’The Feasible Allocation Is’, (s(mins,i),i=1,n)
    PRINT*, ’The Total Cost Is’, tc(mins)
    PRINT*, ’The Average Cost Of This Trial Is’, avertc
    PRINT 78, (ks(j), j=2,n)
78     FORMAT(’Ksj’,913)
    PRINT*, ’CT1, CT2, CT3 Are’, ct1, ct2, ct3
    DO 920 l = 1, ns
        PRINT 81, l, (s(l,m), m=1,n), tc(l)
920    CONTINUE
    STOP
    END

C$INCLUDE URAND.FOR
C
C     FUNCTION SUM
C
FUNCTION SUM(n,x)
  REAL x(20)
  sum = 0.0
  DO 15 j = 1, n
    sum = sum + x(j)
15    CONTINUE
  RETURN
END

C     SUBROUTINE FOR EVALUATION OF ALLOCATION STRING
C
SUBROUTINE COST(n,c1,c3,d,pw,f,r,u,hcs,scs,tc1,ct2,ct3,ks)
  INTEGER i, j, k, l, m, n, x, y, z

INTEGER pw(15), p(15,15), ks(15), is(15)
REAL tc, c1, c2s(15), c3, d, f(15), r(15), u(15,15), hcs(15), scs(15)
REAL fs(15), rs(15), us(15), sts(15), thr, mst, mtts, sumc2s, c2temp
REAL pi(15), v(15), v1(15), lum(15), c(15), us1(15), us2(15), res(15)
REAL ef(15), ief(15), p1, p2, p3, p4, p5, p6, p7, q(15,15), q1(15,15)
REAL sumus, temp2, ct1, ct2, ct3

C
C STEP 1: GENERATE THE ASSIGNMENT VARIABLE MATRIX
C
DO 10 i = 1, N
DO 20 j = 1, N
  IF (pw(i).EQ.j) THEN
    p(i,j) = 1
  ELSE
    p(i,j) = 0
  END IF
20 CONTINUE
10 CONTINUE

C
C STEP 3: CONVERT THE GIVEN INFORMATION PARAMETERS
C
DO 30 j = 1, n
  us(j) = 0
  fs(j) = 0
  rs(j) = 0
DO 40 i = 1, n
  us(j) = us(j) + u(i,j) * p(i,j)
  fs(j) = fs(j) + f(i) * p(i,j)
  rs(j) = rs(j) + r(i) * p(i,j)
40 CONTINUE
30 CONTINUE

C
C STEP 4: DEFINE THE THROUGHPUT RATE
C
DO 50 j = 1, n
  res(j) = rs(j) / (fs(j) + rs(j))
  us1(j) = us(j) * res(j)
  ef(j) = us1(j)
  ief(j) = us(j) * (1 - res(j))
50 CONTINUE

thr = d
DO 60 j = 1, n
  IF (us1(j).LT.thr) THEN
    thr = us1(j)
60 CONTINUE

134
END IF
60 CONTINUE

STEP 5: DEFINE THE STATE SPACE E FOR THE PRODUCTION SYSTEM

DO 70 j=2,n
ks(j)=0
c2s(j)=scs(j)

STEP 6: DEFINE THE TRANSITION MATRIX

DO 75 m=2,20
ks(j)=ks(j)+1
temp=(ef(j-1)+ief(j-1)+ef(j)+ief(j))
p1=ef(j-1)/temp
p2=(ief(j-1)+ief(j))/temp
p3=(ef(j))/temp
p4=p2/(p2+p3)
p5=p3/(p2+p3)
p6=p1/(p1+p2)
p7=p2/(p1+p2)
DO 80 x=1,m
DO 90 y=1,m
q(x,y)=0.0
90 CONTINUE
80 CONTINUE
lum(1)=us(j-1)
lum(m)=us(j)
IF (m.ge.3) THEN
  y=1
  DO 100 x=2,m-1
    q(x,y)=p1
    q(x,y+1)=p2
    q(x,y+2)=p3
    y=y+1
    lum(x)=us(j-1)+us(j)
100 CONTINUE
END IF
q(1,1)=p4
q(1,2)=p5
q(m,m-1)=p6
q(m,m)=p7

STEP 7: DETERMINE THE LIMITING DISTRIBUTION
C
DO 105 k=1,25
   DO 110 x=1,m
      DO 120 y=1,m
         q1(x,y)=0
      120     CONTINUE
   110    CONTINUE
   DO 130 x=1,m
      DO 140 y=1,m
         DO 150 z=1,m
            q1(x,y)=q1(x,y)+q(x,z)*q(z,y)
      150    CONTINUE
   140   CONTINUE
   130  CONTINUE
   DO 160 y=1,m
      v(y)=q1(1,y)
      v1(y)=q1(m,y)
   160  CONTINUE
   IF (v1(1).eq.v1(1).and.v(m).eq.v1(m)) GO TO 320
   DO 180 x=1,m
      DO 190 y=1,m
         q(x,y)=q1(x,y)
      190    CONTINUE
   180  CONTINUE
   105  CONTINUE
320 temp2=0
   DO 200 l=1,m
      temp2=temp2+(v(l)/lum(l))
   200 CONTINUE
   DO 210 l=1,m
      pi(l)=(v(l)/lum(l))/temp2
   210 CONTINUE

C
C STEP 8: DETERMINE THE COST FACTOR FOR EACH STATE
C
DO 215 x=1,m
   DO 220 l=2,m-1
      c(l)=csc(l)
      c(m)=csc(l)*(m-1)
      IF (m.ge.3) THEN
         DO 220 l=2,m-1
            c(l)=csc(l)*(l-1)
      END IF
   220 CONTINUE
END IF
STEP 9: DETERMINE THE EXPECTED COST (C2s) FOR THE
PRODUCTION SYSTEM WITH K KANBANS

   c2temp=0
   DO 230 l=1,m
       c2temp=c2temp+pi(l)*c(l)
   CONTINUE
   IF (c2s(j).LT.c2temp) GO TO 330
   c2s(j)=c2temp
   CONTINUE

STEP 10: DETERMINE THE OPTIMAL NUMBER OF KANBANS

   ks(j)=ks(j)-1
   PRINT*, 'Number of Kanban for Stage j, is', ks(j)
   CONTINUE

STEP 11: DETERMINE THE STARVING TIME OF THE SYSTEM FOR
THE BREAKDOWN OF A WORKSTATION

   DO 240 j=2,n
       IF (us(j).LT.us(j-1)) THEN
           is(j)=ks(j)
       ELSE
           is(j)=0
       END IF
   CONTINUE
   DO 250 i=1,n
       us2(i)=us(i)
   CONTINUE
   DO 260 j=i,n
       IF (us(j).LT.us2(i)) THEN
           us2(i)=us(j)
       END IF
   CONTINUE
   DO 270 j=1,n-1
       sumus=0
       DO 280 l=j,n
           sumus=sumus+(1/us(l))
       CONTINUE
   CONTINUE
   mtt=0.0
   DO 290 l=j+1,n
       mtt=mtt+(is(l)+1)*1/us2(l)
   CONTINUE
290  CONTINUE
       sts(j)=1/rs(j)+sums-mmts
       IF (sts(j).LT.0) THEN
          sts(j)=0
       END IF
270  CONTINUE
       sts(n)=1/rs(n)+1/us(n)

C   STEP 12: DETERMINE THE MEAN STARVING TIME FOR THE
C   PRODUCTION SYSTEM
C
   mst=0.0
   DO 300 j=1,n
       mst=mst+(1-res(j))*sts(j)
300  CONTINUE
C   STEP 13: DETERMINE THE EXPECTED TOTAL COST OF
C   PRODUCTION PER ITEM
C
   sumc2s=0.0
   DO 310 j=2,n
       sumc2s=sumc2s+c2s(j)
310  CONTINUE
   ct1=c1*(1/thr)
   ct2=1/d*sumc2s
   ct3=c3*mst
   tc=ct1+ct2+ct3
C   PRINT*, 'The Total Cost of Production Per Item Is',tc
RETURN
END
Appendix B. Listing of Data File W5.DAT.
Data File W5.DAT

Number of Workstations
5

Operation Cost
240

Shortage Cost of Finished Product
50

Demand Rate
20

Failure Rate
0.25 0.5 0.25 0.1 0.8

Recovery Rate
4 6 4 2 10

Processing Rate
50 20 35 45 24
45 40 30 25 35
35 24 30 20 25
40 45 30 25 35
55 35 25 30 45

WIP Holding Cost
20 20 35 50

WIP Shortage Cost
200 110 155 180
Appendix C. Listing of TC.FOR.
C PROGRAM TOTAL COST
INTEGER i,j,k,l,m,n,x,y,z
INTEGER pw(10),p(10,10),ks(10),is(10)
REAL tc,c1,c2s(10),c3,d,f(10),r(10),u(10,10),hcs(10),scs(10)
REAL fs(10),rs(10),us(10),sts(10),thr,mst,mtts,sumc2s,c2temp
REAL pi(10),v(10),v1(10),lum(10),c(10),us1(10),us2(10),res(10)
REAL ef(10),ief(10),p1,p2,p3,p4,p5,p6,p7,q(10,10),q1(10,10)
REAL sumus,temp2

C DATA INPUT

OPEN(15,FILE='AW2.DAT')
READ(15,*),n,c1,c3,d,(pw(i),i=1,n),(f(i),i=1,n),(r(i),i=1,n)
do 5 i=1,n
  read(15,*),(u(i,j),j=1,n)
5 continue
read(15,*),(hcs(i),i=1,n),(scs(i),i=1,n)
CLOSE(15)

C STEP 1: GENERATE THE ASSIGNMENT VARIABLE MATRIX

DO 10 I = 1, N
  DO 20 J=1,N
    IF (pw(i).eq.j) THEN
      p(i,j)=1
    ELSE
      p(i,j)=0
    ENDIF
  20 continue
10 continue

C STEP 3: CONVERT THE GIVEN INFORMATION PARAMETERS

DO 30 j=1,n
  us(j)=0
  fs(j)=0
  rs(j)=0
  DO 40 i=1,n
    us(j)=us(j)+u(i,j)*p(i,j)
    fs(j)=fs(j)+f(i)*p(i,j)
    rs(j)=rs(j)+r(i)*p(i,j)
  40 continue
30 continue

142
C STEP 4: DEFINE THE THROUGHPUT RATE
   do 50 j=1,n
   res(j)=rs(j)/(fs(j)+rs(j))
   us1(j)=us(j)*res(j)
   ef(j)=us1(j)
   ief(j)=us(j)*(1-res(j))
50   continue
   thr=d
   do 60 j=1,n
   if (us1(j).LT.thr)then
      thr=us1(j)
   end if
60   continue
C
C STEP 5: DEFINE THE STATE SPACE E FOR THE PRODUCTION SYSTEM
   do 70 j=2,n
   ks(j)=0
   c2s(j)=scs(j)
C
C STEP 6: DEFINE THE TRANSITION MATRIX
   DO 75 m=2,10
   ks(j)=ks(j)+1
   temp=(ef(j-1)+ief(j-1)+ef(j)+ief(j))
   p1=ef(j-1)/temp
   p2=(ief(j-1)+ief(j))/temp
   p3=(ef(j))/temp
   p4=p2/(p2+p3)
   p5=p3/(p2+p3)
   p6=p1/(p1+p2)
   p7=p2/(p1+p2)
   do 80 x=1,m
   do 90 y=1,m
   q(x,y)=0.0
80   continue
90   continue
   lum(1)=us(j-1)
   lum(m)=us(j)
   if (m.ge.3) then
      y=1
   do 100 x=2,m-1
      q(x,y)=p1
100  continue
q(x,y+1)=p2
q(x,y+2)=p3
y=y+1
lum(x)=us(j-1)+us(j)
100    continue
end if
q(1,1)=p4
q(1,2)=p5
q(m,m-1)=p6
q(m,m)=p7

C
STEP 7: DETERMINE THE LIMITING DISTRIBUTION
C
do 105 k=1,25
do 110 x=1,m
do 120 y=1,m
   q1(x,y)=0
120    continue
110    continue
do 130 x=1,m
do 140 y=1,m
do 150 z=1,m
   q1(x,y)=q1(x,y)+q(x,z)*q(z,y)
150    continue
140    continue
130    continue
do 160 y=1,m
   v(y)=q1(x,y)
   v1(y)=q(m,y)
160    continue
if (v(1).eq.v1(1).and.v(m).eq.v1(m)) go to 400
do 180 x=1,m
do 190 y=1,m
   q(x,y)=q1(x,y)
190    continue
180    continue
105    continue
400    temp2=0
do 200 l=1,m
   temp2=temp2+(v(l)/lum(l))
200    continue
do 210 l=1,m
   pi(l)=(v(l)/lum(l))/temp2
210    continue
STEP 8: DETERMINE THE COST FACTOR FOR EACH STATE

do 215 x=1,m
    continue
    c(1)=scs(j)
    c(m)=hcs(j)*(m-1)
    if (m.ge.3) then
        do 220 l=2,m-1
            c(l)=hcs(j)*(l-1)
        220    continue
    end if

STEP 9: DETERMINE THE EXPECTED COST (C2s) FOR THE
PRODUCTION SYSTEM WITH K KANBANS

    c2temp=0
    do 230 l=1,m
        c2temp=c2temp+pi(l)*c(l)
    230    continue
    if (c2s(j).lt.c2temp) go to 500
    c2s(j)=c2temp

STEP 10: DETERMINE THE OPTIMAL NUMBER OF KANBANS

    500  ks(j)=ks(j)-1
    print*, 'Number of Kanban for Stage', j, ' is', ks(j)

STEP 11: DETERMINE THE STARVING TIME OF THE SYSTEM FOR
THE BREAKDOWN OF A WORKSTATION

    do 240 j=2,n
    if (us(j).lt.us(j-1)) then
        is(j)=ks(j)
    else
        is(j)=0
    end if
    240    continue
    do 250 i=1,n
        us2(i)=us(i)
        do 260 j=i,n
            if (us(j).lt.us2(i)) then
                260    continue
        250    continue

145
us2(i) = us(j)
end if

260 continue
250 continue
do 270 j = 1, n-1
   sumus = 0
   do 280 l = j, n
      sumus = sumus + (1/us(l))
   end do
280 continue
mtts = 0.0
do 290 l = j + 1, n
   mtts = mtts + (is(l) + 1) * 1/us2(l)
end do
290 continue
sts(j) = 1/rs(j) + sumus - mtts
if (sts(j) .lt. 0) then
   sts(j) = 0
end if
270 continue
sts(n) = 1/rs(n) + 1/us(n)

C
C  STEP 12: DETERMINE THE MEAN STARVING TIME FOR THE
C  PRODUCTION SYSTEM
C
mst = 0.0
do 300 j = 1, n
   mst = mst + (1 - res(j)) * sts(j)
end do
300 continue

C
C  STEP 13: DETERMINE THE EXPECTED TOTAL COST OF
C  PRODUCTION PER ITEM
C
sumc2s = 0.0
do 310 j = 2, n
   sumc2s = sumc2s + c2s(j)
end do
310 continue
print*, 'The c2s for stage j is', c2s(j)
print*, 'The sum of c2s is', sumc2s
tc = c1 * (1/thr) + 1/d * sumc2s + c3 * mst
print*, 'thr=', thr, 'mst=', mst
print*, 'The Total Cost of Production Per Item Is', tc
stop
end
Appendix D. Listing of Data File W10.DAT.
| 0.240 50 20 |
| 0.35 0.25 0.29 0.30 0.5 0.12 0.25 0.45 0.1 0.8 |
| 7 4 3 6 1 4 4 2 5 10 |
| 50 60 20 70 35 25 45 30 24 20 |
| 24 45 35 40 24 30 25 30 30 25 |
| 25 45 60 24 23 45 28 35 24 30 |
| 25 35 27 24 22 30 30 20 25 18 |
| 65 55 40 60 35 45 60 45 50 25 |
| 28 30 28 24 32 25 30 20 25 30 |
| 40 45 45 40 30 35 25 30 25 25 |
| 55 35 25 30 45 25 30 45 60 35 |
| 30 35 45 50 25 35 25 35 30 30 |
| 55 60 45 65 50 50 45 52 48 20 |
| 20 20 35 50 60 50 25 30 35 |
| 200 110 155 180 120 170 135 140 180 |
Appendix E. Listing of Data File W15.DAT.
<table>
<thead>
<tr>
<th>15</th>
<th>240</th>
<th>50</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35</td>
<td>0.25</td>
<td>0.29</td>
<td>0.30</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>40</td>
<td>45</td>
<td>45</td>
<td>40</td>
</tr>
<tr>
<td>24</td>
<td>45</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>28</td>
<td>30</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>45</td>
<td>60</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>28</td>
<td>30</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td>40</td>
<td>45</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>25</td>
<td>45</td>
<td>60</td>
<td>24</td>
</tr>
<tr>
<td>55</td>
<td>35</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>45</td>
<td>50</td>
</tr>
<tr>
<td>55</td>
<td>60</td>
<td>45</td>
<td>65</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>35</td>
<td>50</td>
</tr>
<tr>
<td>200</td>
<td>110</td>
<td>155</td>
<td>180</td>
</tr>
</tbody>
</table>
Appendix F. Listing of Data File W20.DAT.
20 240 50 20
0.35 0.25 0.29 0.30 0.5 0.12 0.25 0.45 0.1 0.8
0.25 0.3 0.2 0.1 0.15 0.5 0.2 0.45 0.35 0.25
7 4 3 6 1 4 4 2 5 10 6 2 4 3 1 7 6 4 3 2
40 65 45 35 50 50 60 20 70 35 25 45 30 24 20 20 35 25 35 50
40 45 40 30 35 25 30 25 25 45 50 35 35 25 35 50 40 40 40 40
24 45 35 40 24 30 50 60 20 35 25 25 30 30 30 35 30 25 40 24 23 22
30 35 45 25 35 25 35 30 30 35 30 25 35 30 30 35 30 25 40 50
25 45 60 50 60 20 70 35 25 24 23 45 35 24 30 25 36 40 20 60
30 35 45 50 25 25 35 30 30 35 30 25 35 30 30 35 30 25 40 50
50 60 20 70 35 25 45 30 24 20 20 35 24 30 25 36 35 25 35 50
28 30 28 24 32 25 20 25 30 28 25 50 60 20 70 35 25 30 22 20
40 45 45 40 30 35 25 30 40 45 50 25 25 25 25 45 50 35 40 40
30 35 45 50 25 35 25 35 30 30 35 30 25 35 30 35 30 25 40 50
25 45 60 24 23 45 35 24 30 25 36 40 20 60 50 60 20 70 35 45
55 35 25 30 45 25 35 24 30 25 36 30 45 60 35 25 35 40 40 50
30 35 45 50 25 25 35 30 30 35 30 25 35 30 30 35 30 25 40 50
50 60 20 70 35 25 45 30 24 20 20 35 24 30 25 36 35 25 35 50
28 30 28 24 32 25 20 25 30 28 25 50 60 20 70 35 25 30 22 20
40 45 45 40 30 35 25 30 25 25 45 50 35 25 35 40 40 40
24 45 35 40 24 30 50 60 20 35 25 25 30 30 30 35 40 24 23 22
30 35 45 30 25 36 35 45 50 25 35 25 35 30 30 35 30 25 40 50
55 60 45 65 50 50 45 52 48 50 35 24 30 25 36 45 55 45 60 45
20 50 60 20 35 25 20 35 50 50 25 35 30 35 50 55 45 25 30
200 110 155 180 120 170 135 140 180 160
150 140 160 180 150 140 130 120 100

152
Appendix G. Listing of SA.FOR.
PROGRAM OF SIMULATED ANNEALING ALGORITHM

INTEGER i,j,k,l,m,n,x,pw(15),ks(20,15),ns,trimin,minks(15)
INTEGER s(20,15),smin(15),relo(15),itemp,cent,mins,kst(15)
INTEGER mu1,mu2,stemp(15),stage,tri,trilim,term,stamin
REAL c1,c3,d,f(15),r(15),u(15,15),hcs(15),scs(15)
REAL rel(15),temp,re(15),vtemp(15),URAND
REAL ttemp,ctl1(15),ct2(15),ct3(15),tc(20),avertc,mintc,last
REAL rand,accept,cp1,cp,cpcon,tricon,smintc,termco
REAL minct1,minct2,minct3

x=15647
termco=1.5
ns=7

C
DATA INPUT
C

OPEN(15,FILE='W10.DAT')
READ(15,*)n,c1,c3,d,(f(i),i=1,n),(r(i),i=1,n)
DO 400 i=1,n
    READ(15,*)u(i,j),j=1,n
400 CONTINUE
READ(15,*)hcs(i),i=2,n),(scs(i),i=2,n)
CLOSE(15)

C
GENERATE ALLOCATION STRINGS
C

DO 410 i=1,n
    rel(i)=r(i)/(f(i)+r(i))
    rele(i)=i
    re(i)=rel(i)
410 CONTINUE

DO 420 k=1,n-1
    DO 430 l=1,n-k
        IF (rel(l+1),LT.rel(l)) THEN
            temp=rel(l)
            rel(l)=rel(l+1)
            rel(l+1)=temp
            itemp=relo(l)
            relo(l)=relo(l+1)
            relo(l+1)=itemp
        END IF
430 CONTINUE
CONTINUE
DO 440 l=1,n
  s(1,l)=relo(l)
  s(2,n+1-l)=relo(l)
CONTINUE
  cent=INT(n/2)+1
DO 445 m=3,4
  s(m,cent)=relo(1)
  DO 450 k=1,cent-1
    s(m,cent-k)=relo(k*2)
  CONTINUE
  DO 460 k=1,n-cent
    s(m,cent+k)=relo(k*2+1)
  CONTINUE
  DO 470 i=1,n
    vtemp(i)=relo(i)
  CONTINUE
  DO 480 i=1,n
    relo(i)=vtemp(n+1-i)
CONTINUE
CONTINUE
DO 490 l=5,ns
  DO 500 m=1,n
    temp=INT(n*URAND(x))+1
    DO 510 i=1,m-1
      x=x+1
      IF (temp.EQ.s(l,i)) GO TO 520
  CONTINUE
  s(l,m)=temp
CONTINUE
tcont2=0
tcont3=0

tcont4=0
DO 530 l=1,ns
    DO 540 m=1,n
        pw(m)=s(l,m)
    CONTINUE
    CALL COST(n,c1,c3,d,pw,f,r,u,hcs,scs,tcx,ctt1,ctt2,ctt3,kst)
    tc(l)=tcx
    ct1(l)=ctt1
    ct2(l)=ctt2
    ct3(l)=ctt3
    DO 545 m=2,n
        ks(l,m)=kst(m)
    CONTINUE
    PRINT*, 'Total Cost For String', l, ' is', tc(l)
    PRINT*, l, ' is', (s(l,m), m=1,n)
CONTINUE

avertc=sum(ns,tc)/ns
PRINT 95,tri,stage,avertc

FORMAT('Average cost of trial', i3, ' in stage', i3, ' is', F13.7)

DEFINE THE BEST INITIAL ALLOCATION STRING

mintc=tc(1)
mins=1
DO 550 l=2,ns
    IF (tc(l).LT.mintc) THEN
        mintc=tc(l)
        mins=l
    END IF
CONTINUE

last=mintc

mint1=ct1(mins)
mint2=ct2(mins)
mint3=ct3(mins)
DO 552 m=1,n

smin(m)=s(mins,m)

CONTINUE
DO 558 m=2,n

minks(m)=ks(mins,m)
CONTINUE
PRINT 93,(s(mins,m),m=1,n),tc(mins)
FORMAT('The Best Allocation Is',1013,' With TC Of',F13.7)

SIMULATED ANNEALING

GENERATE NEW NEIGHBOURHOOD STRINGS (MUTATION)

tri=tri+1
cont1=0
cont2=0
cont3=0
cont4=0
DO 560 l=1,ns
   x=x+1
   mu1=INT(n*URAND(x))/1
   x=x+1
   mu2=INT(n*URAND(x))+1
   IF (mu1.EQ.mu2) GO TO 565
   DO 570 m=1,n
      stemp(m)=s(l,m)
   CONTINUE
   i0temp=stemp(mu1)
   stemp(mu1)=stemp(mu2)
   stemp(mu2)=i0temp

EVALUATE NEW STRING

DO 580 m=1,n
   1->(m)=stemp(m)
CONTINUE
CALL COST(n,c1,c3,d,pw,f,r,u,hcs,scs,tcstemp,co1,co2,co3,kst)
ct1(l)=co1
ct2(l)=co2
ct3(l)=co3
DO 585 m=2,n
   kso(l,m)=kst(m)
CONTINUE

ACCEPTANCE OF NEW STRING

cont1=cont1+1
IF (tcstemp.LE.tc(l)) then
   DO 590 m=1,n
   1->(m}=1->(m)
   1->(m)=stemp(m)
CONTINUE
   1->(m)=1->(m)

590 CONTINUE
\begin{verbatim}
590    s(l,m) = stemp(m)
    CONTINUE
    tc(l) = tc temp
    cont2 = cont2 + 1
ELSE
    accept = EXP((tc(l) - tc temp)/c p)
    x = x + 1
    rand = URAND(x)
    cont3 = cont3 + 1
    IF (rand .LE. accept) THEN
      DO 600 m = 1, n
        s(l,m) = stemp(m)
      CONTINUE
    tc(l) = tc temp
    cont4 = cont4 + 1
  END IF
END IF
560    CONTINUE
C
C DEFINE THE BEST NEW ALLOCATION STRING
C
  aver tc = sum(ns, tc)/ ns
PRINT 91, tri, stage, aver tc
91    FORMAT('Average cost of trial', i3, ' in stage', i3, ' is', F13.7)
  probim = cont2/cont1
  probac = cont4/cont3
  PRINT 99, probim
99    FORMAT('Probability of improve in new string is------', F7.5)
  print 101, probac
101   FORMAT('Prob. of accepting non-improve new string is--', F7.5)
  scont1 = scont1 + cont1
  scont2 = scont2 + cont2
  scont3 = scont3 + cont3
  scont4 = scont4 + cont4
  mintc = tc(1)
  mins = 1
  DO 610 l = 2, ns
    IF (tc(l) .LT. mintc) THEN
      mintc = tc(l)
      mins = l
    END IF
  CONTINUE
610   IF (mintc .LT. last) THEN
    last = mintc
\end{verbatim}
stamin = stage
trimin = tri
minct1 = ct1(mins)
minct2 = ct2(mins)
minct3 = ct3(mins)
DO 615 m = 2, n
    minks(m) = ks(mins, m)
615   CONTINUE
DO 620 m = 1, n
    smin(m) = s(mins, m)
620   CONTINUE
PRINT 97, (smin(m), m = 1, n), minct
97    FORMAT("The Best Allocation Is”, 10I3, ’ With TC Of’, F13.7)
END IF
C
C    UPDATE CONTROL PARAMETER cp
C
    IF (tri.EQ.trilim) THEN
        PRINT *, ’Probability For stage’, stage
        sproim = scont2/scont1
        PRINT *, ’Scont1 and Scont2 are’, scont1, scont2
        print 99, sproim
        sproac = scont4/scont3
        PRINT *, ’Scont3 and Scont4 are’, scont3, scont4
        print 101, sproac
        tcont1 = tcont1 + scont1
        tcont2 = tcont2 + scont2
        tcont3 = tcont3 + scont3
        tcont4 = tcont4 + scont4
        scont1 = 0
        scont2 = 0
        scont3 = 0
        scont4 = 0
        cp = cp*cpcon
        trilim = INT(tricon + tricon*(cp1-cp)+.5)
        itemp = 2*tricon
        IF (trilim.GT.itemp) THEN
            trilim = itemp
        END IF
        tri = 0
        stage = stage + 1
        IF (sminct.EQ.last) THEN
            term = term + 1
            itemp = INT(termco*n+.5)
IF (term.GE.itemp) GO TO 900
ELSE
    smintc=last
term=1
END IF
END IF
GO TO 800

C
C FEASIBLE SOLUTION OBTAINED AND OUTPUT
C
900 PRINT *, 'Probability For The Algorithm'
troim=tcont2/tcont1
troac=tcont4/tcont3
PRINT *, 'Tcont1 and Tcont2 are', tcont1, tcont2
PRINT 99, troim
PRINT *, 'Tcont3 and Tcont4 are', tcont3, tcont4
PRINT 101, troac
PRINT*, 'The Feasible Allocation Is', (smin(i), i=1, n)
PRINT*, 'The Total Cost is', smintc
PRINT 92, stamin, tramin
92 FORMAT('It Is Obtained In Stage ', I3, ', Trial', I3)
PRINT*, 'The Average Cost Of This Trial Is', avertc
PRINT 94, (minks(j), j=2, n)
94 FORMAT('Ksj is', 9I3)
PRINT 96, minct1, minct2, minct3
96 FORMAT('CT1, CT2, CT3 Are', 3F12.6)
DO 920 l=1, ns
    PRINT 98, l, (s(l,m), m=1, n), tc(l)
98 FORMAT('String', I2, ' Is ', 10I2, ' With Total Cost Of', F7.2)
920 CONTINUE
STOP
END

$INCLUDE URAND.FOR

C
FUNCTION SUM

FUNCTION SUM(n,x)
REAL x(20)
sum=0.0
DO 15 j=1,n
    sum=sum+x(j)
15 CONTINUE
RETURN
END
SUBROUTINE FOR EVALUATION OF ALLOCATION STRING

SUBROUTINE COST(n,c1,c3,d,pw,f,r,u,hcs,scs,tc1,ct2,ct3,ks)
INTEGER i,j,k,l,m,n,x,y,z
INTEGER pw(15),p(15,15),ks(15),is(15)
REAL tc,c1,c2s(15),c3,d,f(15),r(15),u(15,15),hcs(15),scs(15)
REAL fs(15),rs(15),us(15),sts(15),thr,mst,mtts,sumc2s,c2temp
REAL pi(15),v(15),vl(15),lum(15),c(15),us1(15),us2(15),res(15)
REAL ef(15),ief(15),p1,p2,p3,p4,p5,p6,p5,q(15,15),q1(15,15)
REAL sumus,temp2,ct1,ct2,ct3

STEP 1: GENERATE THE ASSIGNMENT VARIABLE MATRIX

DO 10 i = 1, N
   DO 20 j = 1, N
      IF (pw(i) .EQ. j) THEN
         p(i,j) = 1
      ELSE
         p(i,j) = 0
      END IF
   CONTINUE
10 CONTINUE

STEP 2: CONVERT THE GIVEN INFORMATION PARAMETERS

DO 30 j = 1, n
   us(j) = 0
   fs(j) = 0
   rs(j) = 0
   DO 40 i = 1, n
      us(j) = us(j) + u(i,j)*p(i,j)
      fs(j) = fs(j) + f(i)*p(i,j)
      rs(j) = rs(j) + r(i)*p(i,j)
   CONTINUE
30 CONTINUE

STEP 4: DEFINE THE THROUGHPUT RATE

DO 50 j = 1, n
   res(j) = rs(j)/(fs(j) + rs(j))
   us1(j) = us(j)*res(j)
   ef(j) = us1(j)
   ief(j) = us(j)*(1-res(j))

   161
50  CONTINUE
    thr=d
    DO 60 j=1,n
      IF (us1(j).LT.thr) THEN
        thr=us1(j)
      END IF
    60  CONTINUE

C
C  STEP 5: DEFINE THE STATE SPACE E FOR THE PRODUCTION SYSTEM
C
DO 70 j=2,n
ks(j)=0
   c2s(j)=scs(j)

C
C  STEP 6: DEFINE THE TRANSITION MATRIX
C
70  CONTINUE
   DO 80 m=2,20
   ks(j)=ks(j)+1
   temp=(ef(j-1)+ief(j-1)+ef(j)+ief(j))
   p1=ef(j-1)/temp
   p2=(ief(j-1)+ief(j))/temp
   p3=(ef(j))/temp
   p4=p2/(p2+p3)
   p5=p3/(p2+p3)
   p6=p1/(p1+p2)
   p7=p2/(p1+p2)
   DO 80 x=1,m
    DO 90 y=1,m
       q(x,y)=0.0
   80  CONTINUE
   DO 90 y=1,m
   90  CONTINUE

80  CONTINUE
   lum(1)=us(j-1)
   lum(m)=us(j)
   IF (m.ge.3) THEN
     y=1
     DO 100 x=2,m-1
        q(x,y)=p1
        q(x,y+1)=p2
        q(x,y+2)=p3
        y=y+1
        lum(x)=us(j-1)+us(j)
   100  CONTINUE
   END IF
   q(1,1)=p4
q(1,2) = p5
q(m,m-1) = p6
q(m,m) = p7

C
STEP 7: DETERMINE THE LIMITING DISTRIBUTION
C
DO 105 k = 1, 25
  DO 110 x = 1, m
    DO 120 y = 1, m
      qi(x,y) = 0
    120    CONTINUE
  CONTINUE
  110 CONTINUE
  DO 130 x = 1, m
    DO 140 y = 1, m
      DO 150 z = 1, m
        qi(x,y) = qi(x,y) + q(x,z)*q(z,y)
      CONTINUE
    140 CONTINUE
  130 CONTINUE
  DO 160 y = 1, m
    v(y) = qi(1,y)
    vl(y) = qi(m,y)
  160 CONTINUE
IF (v(1).eq.v(1).and.v(m).eq.v(m)) GO TO 320
DO 180 x = 1, m
  DO 190 y = 1, m
    qi(x,y) = qi(x,y)
  CONTINUE
  180 CONTINUE
  105 CONTINUE
320 temp2 = 0
  DO 200 l = 1, m
    temp2 = temp2 + (v(l)/lum(l))
  CONTINUE
  200 CONTINUE
  DO 210 l = 1, m
    pi(l) = (v(l)/lum(l))/temp2
  CONTINUE
  210 CONTINUE
C
STEP 8: DETERMINE THE COST FACTOR FOR EACH STATE
C
DO 215 x = 1, m
  c(1) = scs(j)
  c(m) = hcs(j)*(m-1)
  215 CONTINUE
IF (m.ge.3) THEN
   DO 220 l=2,m-1
       c(l)=hcs(j)*(l-1)
   220 CONTINUE
END IF

C
C   STEP 9: DETERMINE THE EXPECTED COST (C2s) FOR THE
C       PRODUCTION SYSTEM WITH K KANBANS
C
   c2temp=0
   DO 230 l=1,m
       c2temp=c2temp+pi(l)*c(l)
   230 CONTINUE
   IF (c2s(j).LT.c2temp) GO TO 330
   c2s(j)=c2temp
   75 CONTINUE

C
C   STEP 10: DETERMINE THE OPTIMAL NUMBER OF KANBANS
C
   330 ks(j)=ks(j)-1
   C       PRINT*, 'Number of Kanban for Stage j, is', ks(j)
   70 CONTINUE

C
C   STEP 11: DETERMINE THE STARVING TIME OF THE SYSTEM FOR
C       THE BREAKDOWN OF A WORKSTATION
C
   DO 240 j=2,n
       IF (us(j).LT.us(j-1)) THEN
           is(j)=ks(j)
       ELSE
           is(j)=0
       END IF
   240 CONTINUE
   DO 250 i=1,n
       us2(i)=us(i)
       DO 260 j=i,n
           IF (us(j).LT.us2(i)) THEN
               us2(i)=us(j)
           END IF
   260 CONTINUE
   250 CONTINUE
   DO 270 j=1,n-1
       sumus=0
       DO 280 l=j,n

   164
sumus = sumus + (1/us(l))

280  CONTINUE
mtts = 0.0
DO 290 l = j + 1, n
   mtt = mtt + (is(l) + 1)*1/us2(l)
290  CONTINUE
sts(j) = 1/rs(j) + sumus - mtt
IF (sts(j).LT.0) THEN
   sts(j) = 0
END IF

270  CONTINUE
sts(n) = 1/rs(n) + 1/us(n)

C
C    STEP 12: DETERMINE THE MEAN STARVING TIME FOR THE
C    PRODUCTION SYSTEM
C
mst = 0.0
DO 300 j = 1, n
   mst = mst + (1-res(j))*sts(j)
300  CONTINUE

C
C    STEP 13: DETERMINE THE EXPECTED TOTAL COST OF
C    PRODUCTION PER ITEM
C
sumc2s = 0.0
DO 310 j = 2, n
   sumc2s = sumc2s + c2s(j)
310  CONTINUE
c1 = c1*(1/thr)
c2 = 1/d*sumc2s
c3 = c3*mst
tc = c1 + c2 + c3

C    PRINT*, 'The Total Cost of Production Per Item Is', tc
RETURN
END
VITA AUCToris

1968  Born in Hong Kong on the 16th of October.

1987  Completed high school education from Niagara Christian
       College, Fort Erie, Ontario, Canada.

1991  Graduated from University of Windsor, Windsor,
       Ontario, with a B.A.Sc. Degree in Industrial
       Engineering.

1994  Currently a candidate for the M.A.Sc. Degree in
       Industrial Engineering at the University of Windsor,
       Windsor, Ontario, Canada.